N. 1

PER IBM E COMPATIBILI MS-DOS

CON DISCO

INTELLIGENZA ARTIFICIALE

by PC USER

PROLOG

LINGUAGGIO NATURALE

SISTEMI ESPERTI

PROGRAMMI INTELLIGENTI

HONEYWELL COURTESY

COMPILATORE PASCAL
SPRITE CAD
LE SUBDIRECTORY SEGRETE
PAC MAN
LE RETI DI COMUNICAZIONE
MS DOS ABC
BASIC LISTING

RIVISTA
PIÙ DISCO
in edicola!

IBM COURTESY

RIVISTA E DISCO PROGRAMMI
PER IBM E COMPATIBILI MS-DOS

SOMMARIO

Direttore
Mario Magrone

Redattore Capo
Sira Rocchi

Direzione Tecnica
Emanuele Dassi

Testi
Carlo Niccolai

Stampa
Garzanti Editore S.p.A.
Cernusco S/N (MI)

NEL DISCO

RICONOSCITORE DEL LINGUAGGIO NATURALE
IL PROGRAMMA CHE DIVIENE SEMPRE PIÙ INTELLIGENTE

Amministrazione, Redazione, Pubblicità, Arcadia srl: C.so Vittorio Emanuele 15, 20122 Milano. Fotocomposizione: Composit. Selezione colori e fotolito: Eurofotolit. Stampa: Garzanti Editore S.p.A. Milano. Distribuzione: SO.DI.P. Angelo Patuzzi spa, Via Zuretti 25, Milano. PC USER è un periodico mensile registrato presso il Tribunale di Milano al n. 31 il 23 gennaio 1987. Resp. Sira Rocchi. Spedizione in abbonamento postale Gr. III/70. Pubblicità inferiore al 70%. Tutti i diritti sono riservati per tutti i paesi. Manoscritti, disegni, fotografie e programmi inviati non si restituiscono anche se non pubblicati. © 1988.

Copertina: Honeywell courtesy.

DATA BASE

LANGUAGE CODES

WORD PROCESSOR

INTELLIGENZA ARTIFICIALE

MS DOS

GRAPHIC CAD

SOFT & HARD TEST

PROGRAMMI APPLICATIVI

UN ANNO DI PROGRAMMI

ABBONATI! SOLO LIRE 111 MILA

PIÙ UN DISCO SPECIAL IN OMAGGIO

UN' OCCASIONE CHE DURA UN ANNO!

Per abbonarsi (ed avere diritto a 11 fascicoli) basta inviare vaglia postale ordinario di lire 111 mila ad Arcadia srl, c.so Vitt. Emanuele 15, 20122 Milano. Fallo subito!

L'INFORMATICA DEL FUTURO

È GIÀ REALTÀ: SI CHIAMA INTELLIGENZA ARTIFICIALE.

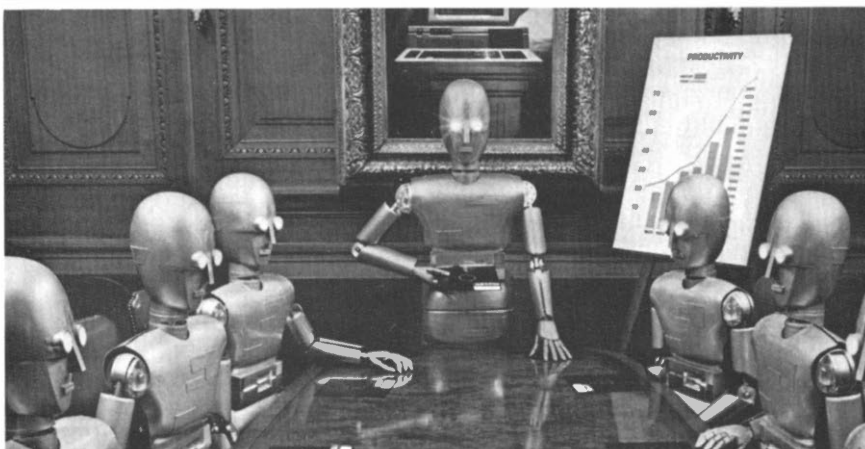
Che cosa è l'intelligenza? Da sola questa domanda ha fatto sì che per secoli intere generazioni di filosofi, psicologi, neurologi pensassero ad enormi quantità di diverse risposte, ciascuna delle quali vera, profonda ma mai completamente esauriente.

Un dizionario definisce l'intelligenza come "La capacità di apprendere fatti e proposizioni, le loro relazioni e di ragionare attorno ad essi". Quale esempio citerei un database relazionale che può memorizzare (apprendere) informazioni, accettare domande (proposizioni) e rappresentare relazioni. In questo modo ciò che esegue un calcolatore: caricare, memorizzare, e accedere alle informazioni, soddisfa alla prima richiesta della definizione di intelligenza.

Fino a poco tempo fa questa tematica è rimasta al di fuori del dominio della tecnologia, solo la fantascienza aveva al proprio interno la nozione di intelligenza applicata alle macchine.

L'uomo, grazie alla propria intelligenza, creatura senza riposo, con una naturale e notevolissima propensione alla costruzione di utensili e strumenti, è riuscito a superare le limitazioni fisiche del corpo umano tramite gli strumenti da lui stesso costruiti: trattori, telescopi, penne, telefoni e migliaia di altri hanno permesso di oltrepassare i limiti imposti dal nostro stesso corpo; non ultimo in ordine di tempo ma alla cima di questa lista di strumenti deve essere posto proprio il calcolatore.

Al calcolatore, così come molte persone lo conoscono e lo amano, non è certo attribuibile il termine di "intelligente", al massimo è una macchina veloce e affidabile, ma incredibilmente stupida, che esegue compiti ben definiti e precisi, senza stancarsi ed in modo ripetitivo. Ciò lo rende senza valore in una società tecnologica come la nostra sempre più schiava della velocità e della precisione; grazie al suo uso siamo stati liberati da una serie di operazioni noiose e ripetitive (da quanto tempo non calcoliamo una radice quadrata con penna e carta?). Ma per quanti sforzi facciano i calcolatori rimangono delle controparti assolutamen-

MAXELL COURTESY

IL GIOCO DELL'IMITAZIONE

Mi propongo di considerare la questione: "Possono pensare le macchine?" Si dovrebbe cominciare col definire il significato dei termini "macchina" e "pensare". Le definizioni potrebbero essere elaborate in modo il più possibile vicino all'uso normale delle parole, ma questo atteggiamento è pericoloso.

Se il significato delle parole "macchina" e "pensare" deve essere trovato esaminando le stesse parole attraverso il loro uso comune è difficile sfuggire alla conclusione che tale significato e la risposta alla domanda "possono pensare le macchine?" vadano ricercati in una indagine statistica del tipo inchieste Gallup. Ciò è assurdo. Invece di tentare una definizione di questo tipo sostituirò la domanda con un'altra, che le è strettamente analoga ed è espressa in termini non troppo ambigui.

La nuova forma del problema può essere descritta nei termini di un gioco, che chiameremo "il gioco dell'imitazione". I giocatori sono tre persone, un uomo (A), una donna (B) e l'interrogante (C), che può essere dell'uno o dell'altro sesso; l'interrogante viene chiuso in una stanza, separato dagli altri due.

Scopo del gioco per l'interrogante è quello di determinare quale delle altre due persone sia l'uomo e quale la donna. Egli le conosce con le etichette X e Y, e alla fine del gioco darà la soluzione "X è A e Y è B" o la soluzione "X è B e Y è A". L'interrogante può far domande di questo tipo ad A e B:
C:"Vuol dirmi X, per favore, la lunghezza dei suoi capelli?"

Ora supponiamo che X sia in effetti A, quindi A deve rispondere. Scopo di A nel gioco è quello di ingannare C e far si che fornisca una indicazione errata, la sua risposta potrebbe essere: "I miei capelli sono tagliati *à la garçonne*, ed i più lunghi sono di circa venticinque centimetri".

Le risposte dovrebbero essere scritte, o meglio ancora, battute a macchina, in modo che il tono di voce non possa aiutare l'interrogante. La soluzione migliore sarebbe quella di avere una telescrivente che mettesse in comunicazione le due stanze. Oppure domande e risposte potrebbero essere ripetute da un intermediario.

Scopo del gioco, per il terzo giocatore (B), è quello di aiutare l'interrogante. La migliore strategia per lei è probabilmente quella di dare risposte veritiere. Lei può anche aggiungere alle sue risposte frasi come "sono io la donna, non dargli ascolto!", ma ciò non approderà a nulla dato che anche l'uomo può fare affermazioni analoghe.

Poniamoci ora la domanda "Cosa accadrebbe se una macchina prendesse il posto di A nel gioco?". L'interrogante darà una risposta errata altrettanto spesso di quando il gioco viene giocato tra un uomo ed una donna? Queste domande sostituiscono quella originale: "possono pensare le macchine?".

da "Computer machinery and Intelligence", Alan Turing, Mind. 50 1950.

te senza ispirazione. Anche se, meglio non ipotecare il futuro, nulla può dirsi di quel che accadrà.

IL COMPUTER: MACCHINA STUPIDA?

Oltre ad essere degli incurabili costruttori di strumenti, gli uomini hanno anche la passione per conoscere e diffondere le informazioni che in un qualsiasi modo arrivano a gestire. Sembra che non esista un limite alla crescita delle conoscenze umane, alcune stime ci dicono che si sta espandendo al ritmo di duecento milioni di parole all'ora.

Sono senza fine gli argomenti su cui l'umanità sta accumulando informazioni, dalle condizioni del tempo, al territorio, alla progettazione dei calcolatori ecc. ecc. ecc., al punto che solo i più eminenti specialisti in ogni campo possono essere considerati senza paura degli esperti.

Torna alla mente la frase che Einstein disse: "Un esperto è quella persona che conosce tutto di niente".

Questo chiaramente crea almeno un problema: abbiamo delle informazioni vorremmo anche usarle. È cosa ovvia l'estrema difficoltà di accedere ad una biblioteca di grandi dimensioni: una persona che volesse cercare una pubblicazione o notizie su un fatto in una grande libreria sarebbe veramente in imbarazzo se non fosse ben organizzata e con molteplici riferimenti incrociati.

Un problema meno ovvio è la continua aggiunta di nuova conoscenza se non si crea un buon sistema di aggiornamento delle notizie presenti e di cancellazione del materiale obsoleto.

TIPI DI CONOSCENZA

La conoscenza di qualsiasi tipo può essere divisa in due grandi classificazioni: Reale ed Euristica; per euristica intendo quella parte di una scienza che si propone di ricercare e di scoprire nuovi fatti e verità.

La conoscenza reale (in inglese Factual) è la più ovvia e necessita di poche attenzioni per essere elaborata; è spesso chiamata "la conoscenza del libro di testo".

La varietà Euristica è d'altra parte un poco più difficile da memorizzare in un calcolatore. È un insieme di intuizioni, associazioni, regole di giudizio, teorie di preferenza, e procedure generali di deduzione (inglese

inference) che, in combinazione con la reale conoscenza di un settore, permettono all'umanità di esibire un comportamento intelligente.

Un ulteriore elemento di viscosità dell'attività del programmatore è il più alto livello di conoscenza che può essere incluso nella categoria Euristica: la "Quasiconoscenza o Metaconoscenza" che è interessata dalle strategie di soluzione dei problemi (problem-solving) e la consapevolezza di come facciamo a pensare.

La conoscenza reale è entrata nei calcolatori ormai da decenni; i sistemi commerciali contengono records dei clienti, del personale, di inventario, ecc., e rendono tipica ma anche banale l'attività della maggior parte dei grandi calcolatori oggi esistenti.

La conoscenza Euristica è molto più difficile da rappresentare in un programma o in un data-base rispetto ai semplici fatti reali; ogni sistema che è inteso come una sofisticata risorsa informativa deve, in un qualche modo, incorporare il livello più alto di conoscenza se non altro che per ridurre in termini semplici e comprensibili il problema di portare qualsiasi parte di informazione alle necessità operative e decisionali del managent.

Supponiamo di avere un sistema creato per fornire ai medici una lista sui sintomi clinici di certe infezioni. Una semplice lista memorizzata in un qualsiasi punto del calcolatore è praticamente senza utilità; in questo caso il solo modo di usare i dati è confrontarli, uno per uno, con i sintomi rilevati.

Il tipo di approccio deve essere rivisto fin dall'inizio: alcuni sintomi sono senza dubbio meno rilevabili di altri il sistema deve avere la capacità di ordinare analisi addizionali (preferendo quelle non invasive), prima di formulare una diagnosi. Debbono essere anche tenute in considerazione l'età del paziente e la sua storia clinica.

Inoltre dobbiamo tenere presente la possibilità di escludere certi fatti se questi sono incompatibili con la diagnosi più probabile.

Se la macchina deve essere un poco più utile di un libro di testo, deve essere capace di fare tutte queste cose, unitamente alla possibilità di aggiornare le proprie informazioni mano a mano che ciò è richiesto. In altre parole deve possedere una sorta di intelligenza.

STORIA DELL'AI

Per i motivi sopra esposti unitamente a molte altre ragioni che vanno dalla inadeguatezza dei metodi standard di programmazione alla pura ricerca, la scienza dei calcolatori ha elaborato una nuova disciplina: l'Intelligenza Artificiale (A.I.).

È difficile stabilire una data di partenza a quello che si usa chiamare AI, comunque il fenomeno che attualmente si identifica con questa allocuzione iniziò intorno al 1960 al MIT quando John McCarthy creò il LISP, il primo linguaggio per l'intelligenza artificiale.

La paternità del termine "intelligenza artificiale" è generalmente attribuita a Marvin Minsky, anche lui del MIT, che nel 1961 scrisse un articolo intitolato "Passi verso l'intelligenza artificiale".

Nella realtà l'intelligenza artificiale non ha niente di nuovo, alcuni dei principi che oggi fanno da base a questa disciplina possono essere datati intorno agli anni '50 quando Alan Turing, elaborò il così detto "gioco delle imitazioni" che è ancora oggi considerato un metodo valido per determinare se una macchina è o meno intelligente.

In pratica il Test di Turing consiste in un interrogatorio che avviene via terminale tra un uomo ed un computer; l'interrogante può tentare ogni modo possibile per determinare se l'interlocutore è un essere umano o una macchina.

A prima vista può sembrare molto facile per l'esaminatore scoprire la differenza ponendo una domanda quale: "Quanto fa 87,8765 diviso per 117?" L'uomo probabilmente attenderà un poco di tempo, mentre il calcolatore risponderà subito un numero con dieci cifre decimali, ma potrebbe anche trovarsi di fronte ad un uomo con una calcolatrice tascabile e ad un programma con un sistema di ritardo per simulare incertezza.

Il lavoro di Turing in questo settore è stato notevolmente profetico, e per i conservatori del 1950, estremamente radicale. Scrisse: "Credo che alla fine del secolo l'uso delle parole e delle opinioni insegnate sarà così profondamente cambiato che uno potrà parlare con macchine pensanti senza pensare di essere contraddetto".

Ancora una volta i progressi tecnologici sono avanti a quanto pensato. Ciò che distingue e rende provocatorie le caratteristiche delle tecnologie della A.I. è che i più nuovi e complessi problemi non trovano limiti.

La maggior parte delle tecnologie, raggiungono la maturità quando i progressi diventano asintotici, gli

sforzi continui ci portano sempre più vicini al limite di ciò che è possibile ma con una velocità sempre minore.

Pensiamo per un attimo ai crescenti sforzi per rendere i dispositivi elettronici sempre più veloci: il tempo che gli elettroni impiegano per muoversi da un punto all'altro del calcolatore rappresenta un limite immutabile, e le future e migliori performances debbono venire da altre idee o tecnologie.

L'Intelligenza Artificiale non sembra avere questo limite, è cresciuta dalle parole profetiche di Turing e da un argomento esoterico di qualche party fino a scienza di specialisti, con congressi internazionali. La sua esistenza è ormai riconosciuta anche fuori dalle accademie, sono certo che in pochi anni i calcolatori che noi conosciamo sono destinati ad essere radicalmente trasformati.

IL LINGUAGGIO ADOTTATO

Nell'accingermi a parlare in modo più esteso dell'intelligenza artificiale e delle applicazioni possibili, mi rendo conto di avere davanti un compito estremamente arduo, sia per le difficoltà intrinseche dell'argomento, che per gli aspetti quasi magici che la stampa a livello divulgativo ha dato alla materia.

Sentii parlare di intelligenza artificiale verso la fine degli anni '70, ascoltavo con un misto di rispetto e invidia quelli che ne parlavano e che venivano guardati come fossero stregoni. Tuttavia dai più la sola definizione di Intelligenza Artificiale, era vista come una brutta malattia che poteva infestare il sano e robusto corpo dell'Informatica Tradizionale.

Ma già allora era ormai chiaro che il tipo di elaborazione tradizionale dei dati o più in generale l'uso dei processi informatici di automatizzazione delle attività; seguendo queste note, con gli esempi che cercherò di presentare e che si andranno ad affiancare ad altre iniziative, sarà possibile al lettore attento confrontare le due metodologie e valutare da solo quale strada usare, o anche solo per valutare la potenzialità di questo nuovo tipo di strumento.

Tra i linguaggi per l'intelligenza artificiale, io ho scelto il PROLOG che ritengo più adatto e più moderno; in questa ottica, anche a costo di qualche critica da parte dei puristi, i programmi e le procedure presentate saranno scritte e provate col Turbo Prolog, prodotto dal costo contenuto, molto diffuso che permette applicazioni di notevolissimo interesse.

QUANDO UN PROGRAMMA È INTELLIGENTE

Scopo finale di questa pubblicazione è primariamente di fornire una serie di dimostrazioni pratiche sull'utilizzo di programmi scritti con linguaggi non formali, ma anche di richiamare l'attenzione dei lettori su queste nuove possibilità informatiche.

Prima che si possa esplorare il regno della I.A. occorre capire cosa significa per un programma essere intelligente e come un programma intelligente differisce da uno non intelligente. Non è un compito facile, anche perché da ogni parte esistono persone che dicono che tutti i programmi sono intelligenti ed altre che spergiurano che nessuno lo è.

Stabiliti i presupposti della prima parte possiamo ora affermare che un database sia intelligente? Questa risposta dipende da cosa si accetta come definizione di ragionamento. Se accettiamo che la manipolazione dei dati di una basedati, come la ricerca, la selezione, l'interrogazione sia un processo di ragionamento, allora il gestore delle basi di dati è un programma intelligente. Ciò porta alla conclusione che molti programmi di calcolatore sono intelligenti.

Questa conclusione è difficile da accettare. Implica che virtualmente tutti i programmi appartengono al campo dell'intelligenza artificiale, una implicazione che non è vera.

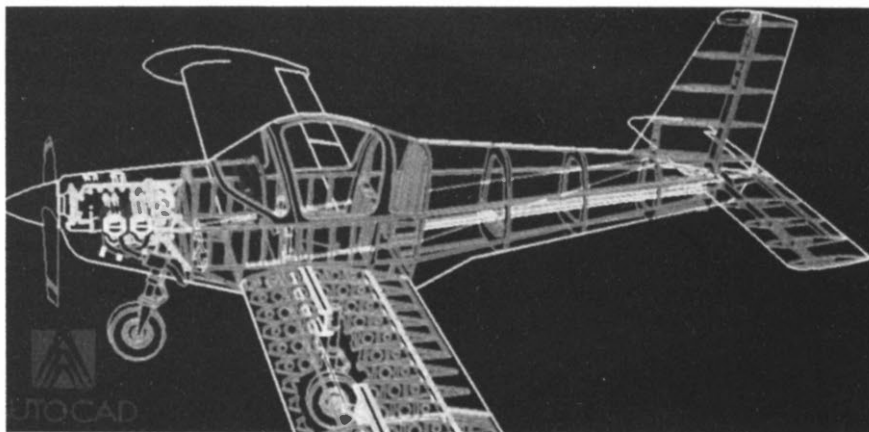
Questo anche perché ciò che un database fa non è conforme a quanto le persone pensano sia un processo intelligente. Ma quando siamo di fronte allo stesso identico lavoro fatto da un uomo, ovviamente pensiamo che esso sia intelligente. Da qui il paradosso: se il programma del database esegue il lavoro allora non pensa, ma se è la persona che lavora allora pensa.

Senza continuare in questi meandri di bassa filosofia, vorrei accettare questa definizione di programma intelligente:
Un programma intelligente si comporta in modo simile ad un essere umano quando confrontato con problemi analoghi. Non è necessario che il programma risolva realmente o tenti di risolvere i problemi nello stesso modo.

Notare che un programma intelligente non deve pensare come un essere umano ma agire come lui (neppure due persone pensano allo stesso modo). Dunque un programma intelligente deve, con gli strumenti che gli sono forniti per interagire col mondo esterno, esibire un comportamento simile all'uomo, mentre un programma non intelligente non può farlo.

Questo potrebbe portare anche a forme aliene di ragionamento, su questo argomento è stato scritto e filosofeggiato così tanto che sarebbe possibile scrivere un intero capitolo dedicato alla fantascienza e alle macchine pensanti.

IL LINGUAGGIO PROLOG

Un argomento base per tutto il mio lavoro è questo: molte persone pensano che Intelligenza Artificiale e grandi calcolatori siano sinonimi. Questo è meno vero oggi che non in passato; tutti i miei programmi sono scritti usando un personal di piccole

dimensioni (portabile) ed il software "TURBO PROLOG" della BORLAND.

Il costo del prodotto TURBO PROLOG è veramente modesto e debbo fare un plauso a questa nota casa internazionale per avere così inaspettatamente diffuso un prodotto ritenuto ingiustamente astruso e difficile.

Il PROLOG è un linguaggio per calcolatori con peculiarità e aspetti che lo rendono differente da tutti gli altri linguaggi quali il PL/1, il COBOL, il PASCAL, il BASIC, ecc.; la differenza è più profonda che la semplice diversità della sintassi, è nel vero nucleo del linguaggio: ciò che il programma significa.

Un programma Prolog può essere visto come un insieme di assioni logici ove l'esecuzione del PROLOG stabilisce una prova che certe desiderate conclusioni derivano dall'insieme dei suoi assiomi.

Il nome Prolog sta per PROgramming in Logic ed è stato inventato nei primi anni del 1970 quale pratica implementazione di un dimostratore per un teorema di un sottoinsieme di logica del primo ordine.

Visto come la realizzazione pratica di un linguaggio per le applicazioni dell'intelligenza artificiale, il PROLOG incorpora molte delle caratteristiche di altri linguaggi, quali il LISP, e si occupa primariamente di strutture ad albero di cui le liste del LISP sono un sottoinsieme; i programmi scritti in PROLOG sono normalmente ricorsivi.

AREE DELL'AI

Il campo di ricerca dell'AI è composto da molte aree di studio, quelle più comuni e più importanti sono:
- Ricerca delle soluzioni.
- Sistemi Esperti.
- Elaborazione del linguaggio naturale.
- Robotica.
- Apprendimento.
- Logica.
- Probabilistica e Logica casuale.

Alcune di queste aree rappresentano la soluzione finale del lavoro, ad esempio i sistemi esperti, altri come la ricerca di soluzioni o l'elaborazione del linguaggio naturale sono spesso aggiunti ad altri programmi per migliorarne le prestazioni. ■

UNA MINIERA DI PROGRAMMI

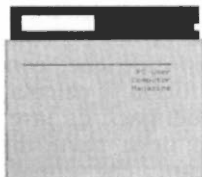
NEI FASCICOLI ARRETRATI DI

PC USER

PC USER 1 - Pianoforte elettronico, Utility per modem, Dos Utility, Scacchi, Convertitore DBIII - Clipper

PC USER 2 - Paratrooper, Copiatore, Spooler per stampante, Black Jack, Utility in C.

PC USER 3 - Basic interprete, Utility musicale, Spacewar, Turbo Pascal utility, Utility disco.

PC USER 4 - 3D Pac Man, Compilatore Pascal, Sprite designer, Dos utility, Turbo Pascal programs.

PC USER 5 - Sailing, Archivio intelligente in DBIII, Menu per DBIII, Disk edit.

PC USER 6 - Analizer per Basic, Tron, Disco-ram, Archivio minerali in DBIII, Agenda telefonica in Lotus.

PC USER 7 - Datab quasi il DBIII, Arc, DBIII Flow, Abaco, Framework money.

PC USER 8 - Database generator, Totocalc, Turbo Pascal form, Grafici a curve in DBIII, Basic Val extension, Editor per file ASCII.

PC USER 9 - Emulatore CGA per Hercules, Soft-in, Cataloga, Fatture, Turbo Prolog example, Procedure in DBIII, Codice fiscale.

PC USER 10 - Word processor, Clipper windows, Assemblatore, Mini expert system, Tabel, Assembler utility.

PC USER 11 - Generatore programmi in DBIII, Organizer per hard disk, Elaboratore di testi, CAD-3D, Nim game, Generatore di programmi Basic.

PC USER 12 - Indentatore per DBIII, Agenda onnipresente, Editor per schermate di presentazione, DBIII Talk, Utility per Turbo Pascal, Conto corrente in Lotus.

PC USER 13 - File Manager Archivio Superinterattivo, Maxi Paint, Sintetizzatore vocale, Ricerca fonetica in DB III.

CHIEDI IL FASCICOLO CHE TI MANCA
inviando un vaglia postale ordinario di lire 15.000 ad Arcadia, Vitt. Emanuele 15, Milano 20122.

IL PROLOG STRUMENTO PRINCIPE

PROGRAMMAZIONE, ESECUZIONE E CONTROLLO DEL LINGUAGGIO DEDICATO ALLE APPLICAZIONI DELL'INTELLIGENZA ARTIFICIALE.

Senza pensare di fare un trattato sul Prolog, seguendo gli esempi e le note di questo capitolo, sarà possibile al lettore attento capire qualsiasi programma, e tentare anche qualche cosa con il manuale dell'interprete.

Ho letto da molte parti come il linguaggio Prolog sia utile per risolvere alcuni problemi complessi come l'analisi del hardware e del software, così come gli aspetti dell'ingegneria del software almeno per le problematiche meno complesse. Molti hanno sottolineato l'utilità del Prolog per l'implementazione di Sistemi Esperti; alcune variazioni al linguaggio, già disponibili sul mercato, lo rendono molto utile anche per applicazioni diverse dalla A.I.

Con riferimento a quanto detto circa la nascita del Prolog si pone una importante domanda: è corretta la relazione tra la dimostrazione dei teoremi e la programmazione? Nei fatti esiste una stretta relazione, il Prolog è caratterizzato da una visione procedurale (o di programmazione) e una dichiarativa (o logica) del programma, da ciò la elaborazione delle informazioni con il Prolog ricava il proprio valore.

Questa relazione può essere illustrata comparando in modo critico gli aspetti significativi di un programma procedurale con quello di un programma logico. La parte significativa di un programma PASCAL è contenuta nella sequenza dei passi che il calcolatore esegue con un certo input fornito. L'input, l'output ed il flusso procedurale sono tra loro mescolati e non possono essere separati.

Il programma PROLOG incorpora due chiari e distinti concetti. Uno è il procedurale che, come il PASCAL, informa il calcolatore sulla sequenza dei passi da fare, il secondo, quello della parte dichiarativa non ha niente a che vedere con il calcolatore. Un programma Prolog può essere letto come una serie di statement (Chiamati ASSIOMI o CLAUSOLE) che definiscono aspetti del mondo reale e possono essere sia vere che false.

È opportuno presentare una definizione della parola ASSIOMA: "Sentenza cui si conferisce un valore assoluto, principio che non ha bisogno di prova o dimostrazione, proposizione che si ammette universalmente"; tutto questo secondo un noto dizionario filosofico.

LA PROGRAMMAZIONE

L'esecuzione di un programma Prolog, da questo punto di vista, è la somma delle visioni logiche per una certa clausola e che può essere dedotta dalle clausole logiche che costruiscono il programma. La realizzazione di una prova della clausola obiettivo (goal) dimostra che l'obiettivo deriva logicamente dagli assiomi che sono nel programma Prolog. È questa l'essenza fondamentale del Prolog.

Ma come si scrive un programma Prolog? Consideriamo per primo questo paragrafo in normale italiano:

"La città di Firenze prende la sua acqua da una notevole serie di sorgenti; una di queste è collocata sugli Appennini ed è il bacino artificiale della Penna. Questa riserva, di quando in quando, si rende indisponibile, si asciuga; in questi casi è completamente vuota. Quando una città riceve la sua acqua da una riserva esaurita, allora deve razionare l'uso dell'acqua".

Questo paragrafo esprime una varietà di pensieri, al punto che una persona dopo averlo letto è in condizione di poter rispondere a qualche domanda tra le quali:
- La città di Firenze deve razionare l'uso dell'acqua?
- Da dove la città di Firenze prende la propria acqua?
- Quando una città deve razionare l'uso dell'acqua?

Il contenuto logico semplificato del paragrafo può essere espresso nelle seguenti tre frasi.
a) La città di Firenze prende la propria acqua dal bacino della Penna.
b) Il bacino della Penna è vuoto.
c) Una città deve razionalizzare l'uso dell'acqua se prende la propria acqua da riserve e Una di queste è vuota.

La U dell'articolo Una è maiuscola perché rappresenta una variabile ed in Prolog iniziano con la lettera maiuscola.

Vi prego di notare che le prime

due frasi esprimono fatti semplici, mentre l'ultima frase esprime una verità condizionata. Quando espressa in Prolog, ogni frase è chiamata CLAUSOLA. Le condizioni che seguono la parola "se" sono chiamate CORPO della clausola. Le prime due frasi non hanno CORPO, in questo caso si dice che il CORPO è vuoto.

Esistono molti modi di formalizzare queste frasi nella logica formale. Il grado di complessità della formalizzazione dipende dalla domanda cui ci si aspetta di dover rispondere. Al minor livello di dettaglio, per esempio, la prima frase può essere semplicemente considerata una clausola senza generalizzazione né struttura. Potrebbe essere solo usata per rispondere alla domanda: "Firenze prende la sua acqua dal bacino della Penna?".

Al maggior livello di dettaglio, la frase può essere considerata l'espressione di una varietà di cose. Esistono due obiettivi, una riserva (la Penna) ed una città (Firenze). Esiste un altro obiettivo (l'acqua) ed una relazione (prende) ciò dice una cosa (Firenze) prende un'altra cosa (l'acqua) da una terza cosa (la Penna).

Il caso più frequente si colloca tra il minore ed il maggior dettaglio. In questo esempio, consideriamo due oggetti, il bacino della Penna e la città di Firenze, e rendiamo "prende acqua da" la relazione tra questi due oggetti. Questa relazione può essere usata per esprimere altri fatti, come "da dove Torino prende la propria acqua".

La seconda frase, circa il razionamento dell'uso dell'acqua, può essere rappresentata con strutture e programmi in Pascal, ma il linguaggio Prolog permette particolari e semplici rappresentazioni di queste clausole. La forma normale di una relazione Prolog è chiamata PREDICATO ed è espressa come segue:
<nome della relazione> (<lista degli obiettivi da relazionare>)

Come in altri linguaggi, le variabili debbono essere distinte dalle costanti e dalle norme di relazione. In Prolog, ed in questi esempi, le variabili iniziano con una lettera maiuscola. Le parole "if" e "and" possono essere rappresentate da :— e &. Le lettere A, B e C sono predicati arbitrari, la forma generale di una clausola Prolog è come segue:
A :— B & C
Che si legge "A è vero se B e C sono veri". Come detto in precedenza, le condizioni (B & C) sono chiamate corpo della clausola, la conclusione (A) è chiamata TESTA. In un corpo possono esistere un qualsiasi numero di condizioni.

Le clausole seguenti in Prolog corrispondono alle precedenti frasi:
1— vuoto(bacino(penna)).
2— città_prende_acqua_dal_bacino (firenze, penna).
3— deve_razionare_acqua_uso(Una_città)
:—città_prende_acqua_dal_bacino (Una_città,Una_riserva)
& vuota(riserva(Una_riserva)).

In aggiunta alla normale forma, molte implementazioni del Prolog permettono un addolcimento sintattico; cioè si possono usare queste frasi più vicine ad un linguaggio parlato
1— la_riserva_della_penna è vuota.
2— firenze prende la_propria_acqua dalla riserva_della_penna.
3— Una_città deve razionalizzare l'uso_dell'acqua se Una_città prende la_propria_acqua da Una_riserva e Una_riserva è vuota

Si chiama microExplorer la nuova stazione di sviluppo nel campo dell'intelligenza artificiale. Incorpora il Lisp chip della Texas Instruments installato in un Macintosh II.

NESSUNA INTELLIGENZA INTRINSECA

Il linguaggio Prolog ha una grande libertà di scrittura, leggendo le clausole di alcuni programmi sembra di trovarci davanti a normali frasi scritte in pieno italiano, ma at-

PER PROGRAMMARE IN LISP...

Il Prolog, e più precisamente per i sistemi MS-DOS il Turbo Prolog, non è l'unico linguaggio dedicato per le applicazioni d'intelligenza artificiale. Vi è anche il Lisp, anch'esso un linguaggio deduttivo nato prima del Prolog. Volendo programmare in questo linguaggio sotto i sistemi MS-DOS vi è il Lisp della Microsoft che con l'ultima versione, la 5.1, ha raggiunto un alto livello di prestazioni. Il Lisp della Microsoft ha oltre 400 istruzioni del Common Lisp in modo da renderlo altamente compatibile con le applicazioni realizzate sui mini computer. Può lavorare anche con soli 128K Ram ed un solo drive. Incorpora precise funzioni matematiche e di debugging. Inoltre consente la creazione di finestre. Per informazioni: Microsoft Spa, Cologno Monzese (MI), tel. 02/2549741.

TERMINOLOGIA USATA NEI PROGRAMMI PROLOG (TURBO PROLOG)

Variabile (Variable): Un nome che inizia con una lettera *maiuscola* che può essere usata per rappresentare il valore (possibilmente non conosciuto) di certi oggetti.
Oggetto (Object): Il nome di un elemento individuale di un certo tipo. Nella frase: Piace (Giovanni, Maria) la relazione è "Piace" mentre gli oggetti sono Giovanni e Maria. I nomi sono liberi ma debbono essere sottoposti a queste regole: iniziare con una lettera maiuscola seguita da un qualsiasi numero di lettere, numeri o "underscore (sottolineatura) _". Le relazioni possono essere ogni combinazione di lettere, numeri o sottolineature. Sono valide relazioni del tipo:
possiede(Carlo,Cavallo)
macchina(fiat,gialla,berlina, turbo_diesel)
che significano:
Carlo possiede un cavallo e
La macchina è una fiat berlina turbo diesel gialla
Dominio (Domain): Indica il tipo dei valori che possono assumere le variabili oggetto delle relazioni. Es.: Interi, simboli, numeriche reali, ecc.
Obiettivo (Goal): L'insieme delle relazioni, possibilmente coinvolgenti oggetti e variabili, cui il Prolog deve tentare di soddisfare.
Albero degli obiettivi (Goal tree): Una rappresentazione diagrammatica delle possibili scelte che possono essere fatte durante la valutazione dei sotto obiettivi che costituiscono il GOAL.
Variabile anonima: La variabile "_" usata al posto di una normale variabile quando non serve assegnare il valore.
argomenti: Nome collettivo per gli oggetti e le variabili nelle relazioni.
Backtracking: Il meccanismo costruito dal Turbo Prolog con cui, quando la valutazione di un sotto-obiettivo è completa, il turbo Prolog torna al precedente sotto-obiettivo e cerca di soddisfarlo in altro modo.
Chiamata ad un sotto-obiettivo (o predicato). Un'espressione che ci dice che il Turbo Prolog sta tentando di soddisfare un certo sottobiettivo.
Clausola: Un fatto od una regola di un particolare predicato seguito da un punto.
Cut, taglio (o !): Il comando dice al Turbo Prolog di fare tutte le scelte per valutare il predicato che contiene il ! . Nel caso di fallimento del subgoal il ! impedisce il backtracking.
Fatto: Una relazione tra due oggetti.
Fail: Un sotto-obiettivo che il Turbo Prolog non può soddisfare.
Variabile libera (free variable): Una variabile che non si riferisce a nessun valore.
Lista (List): Un insieme speciale di oggetti che consistono di elementi chiusi in parentesi quadre e separati da virgole.
Regola (Rule): Una relazione tra un fatto ed una lista di sotto-obiettivi che debbono essere soddisfatti affinché il fatto sia vero.
Predicato Standard (standard predicate): Un predicato già definito internamente nel Turbo Prolog.
Stringa: Un numero arbitrario di caratteri chiusi da una coppia di doppi apici (virgolette).

tenzione il significato dei simboli dipende dall'analista umano e non dal calcolatore. Il Prolog non può fare giudizi associativi circa il contenuto o il significato dei simboli usati nel programma, e non può dedurre altro se non quello che è strettamente fornito. Non ha niente del generale e comune senso della conoscenza che una persona ha.

Dunque nessuna intelligenza intrinseca, ma solo quella associata alla conoscenza dell'uomo che ha fornito il valore ai simboli e "all'esperienza" associata ai fatti contenuti nelle basi di dati.

ESECUZIONE DI UN PROGRAMMA

Fino ad ora ho mostrato come appare un programma Prolog e come viene letto, ora cercherò di illustrare come viene eseguito. Ogni relazione in Prolog è una procedura; nell'esempio del razionamento dell'acqua ci sono tre procedure:
"vuoto,"
"città_prende_acqua_dalla_riserva,"
"bisogna_razionare_l'uso_dell'acqua"

In questo esempio è presente una sola clausola per procedura, ma potrebbe essercene di più, con quanto detto sopra si potrebbe anche sapere che Torino prende l'acqua dalla riserva del fiume Po:
città_prende_acqua_dalla_riserva (po,fiume).

Ora abbiamo due clausole per la procedura
"città_prende_acqua_da_riserva"

Considerando il mondo intero, ce ne potrebbero essere decine di migliaia.

In Prolog una procedura è anche conosciuta come un obiettivo (goal) che deve essere verificato.

La forma più generale di un programma Prolog è costituita da una successione di termini, detti anche FATTI, e di REGOLE; collettivamente fatti e regole vengono chiamate CLAUSOLE.
Il corpo di una clausola è una sequenza di procedure chiamate o invocate; il corpo della sola clausola nella procedura per
"deve_razionare_l'uso_dell'acqua"
è una sequenza di due procedure chiamate,
"città_prende_acqua_dalla_riserva"
e "vuota".

Le procedure del Prolog contengono solo chiamate ad altre procedure.

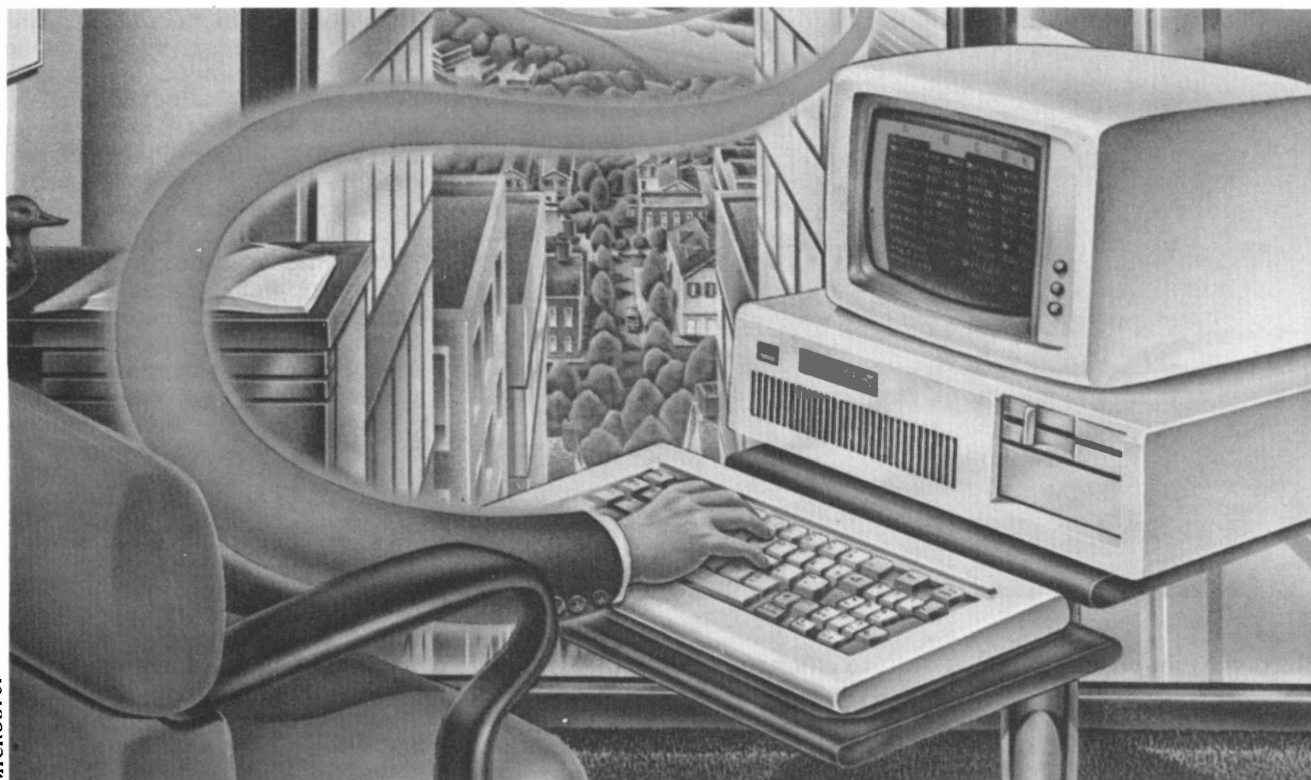
Le regole vengono trattate come segue:

L'interprete cerca di soddisfare un goal contenuto in una domanda esaminando le clausole del programma nell'ordine in cui sono riportate.

La chiamata ad una procedura Prolog può avere successo o fallire. Se ha successo viene confermato un tentativo per un caso dell'obiettivo (goal); se fallisce non è confermato alcun tentativo.

ASSEGNAZIONE DELLE VARIABILI

Il Prolog non ha funzioni di assegnazione delle variabili, adopera invece un processo di accoppiamento

MICROSTUF

dei modelli, chiamato unificazione, per dare valore alle variabili. Quando una procedura è chiamata, ogni clausola della procedura è a turno provata. Le variabili e le strutture dei dati nel goal sono confrontate (matched) con quelle nella testata della clausola. Una variabile si accoppia con tutti, mentre una costante si accoppia solo con la stessa costante

Nell'esempio seguente il goal:
città_prende_l'acqua_dalla_riserva (F,penna)
si accoppia con:
città_prende_l'acqua_dalla_riserva (firenze,penna)

In questo processo di accoppiamento la variabile F assume il valore firenze.

Quando si verifica che una clausola si accoppia con il goal, l'elaborazione è ripetuta su tutti i goal del corpo. Se tutti hanno successo, ha successo anche il goal che si accoppia con la testa della sua clausola. Il risultato ottenuto è illustrato da questo esempio.

Supponete che il goal sia questo:
città_prende_acqua_da_riserva (firenze,P)
Nella procedura esistono queste due clausole:
città_prende_acqua_da_riserva (firenze,penna)
città_prende_acqua_da_riserva (torino,po)

Provando la prima clausola, la costante "firenze" si accoppia con sè stessa nel goal e nella testa. La variabile R nel goal allora si accoppia con penna; e dato che questa clausola non ha condizioni da provare, il goal ha immediatamente successo.

Supponete che il goal sia:
città_prende_acqua_da_riserva (torino,P)

L'accoppiamento di "firenze" con "torino" fallisce, ciò provoca che sia esaminata la seconda clausola, questa ha successo accoppiando la variabile del goal P con "po".

Dove il goal consiste di una sequenza di chiamate alle procedure, un subgoal può avere successo mentre il susseguente subgal fallisce. In questo caso, avviene un backtracking, a causa di una soluzione alternativa al precedente successo del subgoal.

Sebbene non abbia parlato se non sommariamente del backtracking, è sufficiente capire che, in ordine al successo del goal, tutte le chiamate alle procedure nel corpo di una clausola debbono avere successo.

Se il goal è
deve_razionare_l'uso_dell'acqua (Una_città)
viene chiamata la corrispondente regola, ricordatevi che, come ho detto, una variabile inizia con una lettera maiuscola, risulta chiaro che "Una_città" non ha un valore.

La regola ha due sobgoal nel suo corpo; ciascuno di essi deve essere soddisfatto in turno come segue:
città_prende_acqua_dalla_riserva (Una_città, Una riserva) & vuota(riserva(Una_riserva)).

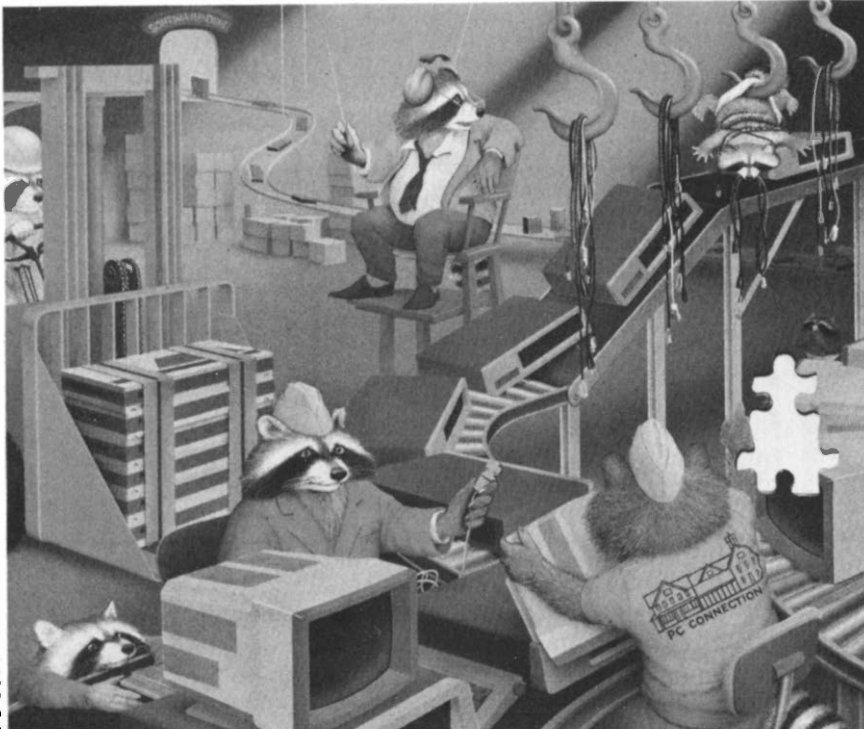
Prima viene chiamata la procedura "città_prende_acqua_dalla_riserva". Poi è chiamata la procedura "vuota".

Supponiamo che sia posto il goal
città_prende_acqua_dalla_riserva (Una_città, Una riserva)
prima si accoppia con la clausola
città_prende_acqua_dalla_riserva (torino,po)
Ora hanno un valore le variabili:
Una_città = torino
Una_riserva = po

Il successivo obiettivo è
Vuota(riserva(po))

Dagli elementi presenti nel programma non può essere provato e la chiamata fallisce, ciò causa backtracking. Il goal precedente ha questa soluzione che si accoppia con la clausola:
città_prende_acqua_dalla_riserva (firenze,penna)

Pertanto le variabili assumono il seguente valore:
Una_città = firenze
Una_riserva=penna
questa volta il subgoal "vuoto(riserva(penna))"ha successo ed abbiamo quanto segue dato che tutti e due gli

PC CONNECTION

obiettivi hanno successo:
dobbiamo_razionare_l'uso_dell'acqua(firenze) poiché
città_prende_acqua_dalla_riserva (firenze,penna) &
vuota (riserva(penna))

Il Prolog permette degli sviluppi con strutture ad albero, come variabili e costanti, tutto questo rende l'elaborazione di unificazione molto potente.

LOGICA E CONTROLLO

Ho cercato di illustrare un modo di leggere un programma Prolog come una serie di frasi logiche ed un altro modo di leggerlo come una serie di procedure chiamate o invocate.

È il contenuto significativo logico che rende diverso il modo di lavorare con il Prolog.

L'autore di un programma Prolog può cambiare durante il lavoro questo modo di vedere il programma, se usato correttamente rende più facili alcune attività; per esempio dal punto di vista dichiarativo la correttezza di un programma Prolog è indipendente dal metodo particolare usato per eseguire una prova.

Ciò significa che il programmatore non deve interessarsi del controllo del flusso di lavoro del programma per stabilire la correttezza dei risultati. Il programmatore deve comunque porre attenzione agli aspetti di controllo per migliorare l'efficienza.

Mi preme sottolineare che l'aspetto efficienza del programma eseguibile è il solo aspetto che il programmatore Prolog deve curare, occorre una buona dose di esperienza per non incorrere in delusioni di performance per programmi di una certa dimensione.

Il precedente esempio mostra che il contenuto dichiarativo del programma è corretto quando risponde: Si, Firenze deve razionalizzare l'uso dell'acqua.

Questo è vero senza riguardo sia per l'ordine in cui le subprocedure sono chiamate che per l'ordine in cui le frasi appaiono nei programmi.

Naturalmente è possibile non essere d'accordo con i dati assunti, in questo caso si può dissentire dalle conclusioni, ma se si accetta che le assunzioni siano vere, allora si può concludere che anche le conclusioni sono vere.

Il programma proposto (molto semplice, vero?) può rispondere a domande circa le riserve vuote (quale riserva è piena?) e sulla fornitura di acqua alle città (quale città prende la sua acqua e da quale riserva), e non è circoscritto alla domanda se la città di Firenze deve limitare l'uso dell'acqua.

Proseguendo in questa attività vi accorgerete spesso che il programma può spesso rispondere anche a domande non pensate, naturalmente sempre e solo con quel poco di esperienze e conoscenza di cui VOI lo avete dotato.

I programmi logici sono più generali in questo senso che i programmi procedurali, ma vi accorgerete che sono anche molto modulari, nel senso che ogni cosa è scritta come molte piccole procedure da eseguirsi in modo non prestabilito.

Queste procedure possono essere anche usate in basi di dati di tipo relazionale.
La procedura unica
"prende_l'acqua_dalla_riserva"
può dire sia da quale riserva la città di Firenze prende l'acqua, o quale città prende l'acqua dalla riserva della Penna, e così via. Le variabili di input e di output possono variare da una chiamata alla successiva.

Questa proprietà di un programma Prolog chiamata "reversibilità relazionale", è spesso utile nel ridurre il numero delle righe di codice da scrivere.

In Pascal, per esempio, debbono essere scritte delle funzioni separate per trovare le riserve delle città e per trovare le città che usano delle riserve, avete invece già intuito che in Prolog esiste solo una procedura relazionale.

STRUTTURA DEL TURBO PROLOG

Un programma scritto in Turbo Prolog ha una struttura molto semplice ma vincolante, i piccoli vincoli sintattici imposti dal prodotto sono ampiamente ripagati da una elevata efficienza raramente riscontrabile in altri prodotti presenti sul mercato.

Nel Turbo Prolog occorre definire i tipi, cioè una serie di costanti, detti anche dominio (domain). Per ogni argomento di ciascun predicato usato nel programma, si deve specificare il dominio che contiene le costanti che possono comparire al posto di quell'argomento. Significa elencare tutti i domini usati, e, per ogni predicato, indicare a quale dominio appartengono le costanti che possono comparire in ciascuno dei suoi argomenti. Il programma risulta formato da almeno tre sezioni.

```
domains
        città = symbol
        quanta_acqua = real
predicates
          vuoto(città,quanta_acqua)
clauses
        riserva(firenze,penna)
        riserva(torino,po)
```

La sezione goal in questo esempio non è specificata, verrà chiesta dall'interprete all'inizio del lavoro.

■

I PROGRAMMI

UNA COLLEZIONE DI PROGRAMMI SUI TEMI DELL'INTELLIGENZA ARTIFICIALE. VEDIAMO QUALI OPERAZIONI BISOGNA ESEGUIRE PER POTERLI CARICARE NEL COMPUTER E COME USARLI.

I programmi registrati sul disco funzionano con qualsiasi sistema MS-DOS (PC, XT, AT IBM e compatibili). Per vedere i files contenuti nel disco bisogna accendere il computer e, dopo aver caricato il sistema operativo, inserire nel drive A il disco dell'IA. Quindi posizionarsi con il prompt in A > (digitare A: seguito da return).

Per caricare i programmi digitare START e seguire le istruzioni a video.

ATTENZIONE: affinché i programmi funzionino correttamente è necessario avere nel sistema operativo il file CONFIG.SYS così configurato:

FILES=20
BUFFERS=20

Se non dovesse esserci bisogna crearlo usando un comune editor oppure direttamente da MS-DOS con il comando copy con: config.SYS.

IL SISTEMA ESPERTO

Molte applicazioni dell'intelligenza artificiale mirano ad ottenere dei programmi capaci di aiutare l'uomo nello studio e nella realizzazione di soluzioni inerenti particolari problemi.

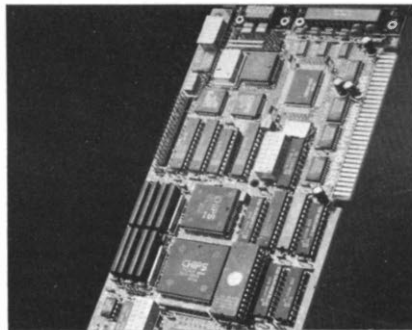
Questi progetti dell'informatica sono conosciuti come "sistemi esperti". Il programma qui descritto rappresenta un esempio di sistema esperto facile da usare ed espandibile. Un sistema esperto (vedi articolo) è costituito da una base dati la quale definisce la conoscenza ad un motore inferenziale capace di utilizzare la base dati per trarne delle deduzioni. Su disco sono registrati due programmi: il sistema esperto vero e proprio (scritto in Turbo Prolog) e il gestore della base dati dal quale è possibile attivare il sistema esperto predisponendogli la base dati. È proprio da quest'ultimo programma che andremo ad esaminare il sistema esperto digitando il nome CIBER00.

Caricato il programma apparirà un menu composto da cinque opzioni siglate dalla lettera A alla lettera E. Il programma CIBER00 serve, come dicevamo, ad aggiornare i dati utilizzati dal sistema esperto, in altre parole si tratta di un programma di archivio. La base dati registrata su

disco comprende già alcuni dati che andremo ad esaminare più avanti. Ora dobbiamo soltanto conoscere la funzione del sistema esperto il quale viene chiamato digitando CIBER51 o attivato da CIBER00. La funzione del sistema esperto è quella di dedurre quale oggetto stiamo pensando. Esso quindi procederà a farci diverse domande circa gli oggetti che il sistema esperto conosce. A seconda delle risposte (si/no) che noi daremo il computer sarà in grado di dedurre l'oggetto pensato. Naturalmente il sistema esperto dovrà avere nella sua conoscenza, ovvero nella sua base dati, la descrizione dell'oggetto da scoprire altrimenti non potrà avanzare nessuna tesi. Ogni oggetto compreso nella base dati è descritto attraverso degli attributi. Più la base dati è grande, cioè più oggetti sono stati definiti, e maggiore è la conoscenza e l'esperienza del programma. Naturalmente un sistema esperto è tanto preciso quanto più sono gli attributi che definiscono ogni singolo oggetto.

Nel disco sono registrati i seguenti oggetti definiti dagli attributi riportati nelle parentesi:
calcolatrice (è_piccola_e_tascabile, ha_tasti, visore_lcd) monitor (cavo_alimentazione, cavo_collegamento_computer, schermo, visualizza_caratteri_e_grafici)
radio (altoparlante, riceve_via_etere, si_ascolta)
tastiera (cavo_collegamento_computer, possiede_tasti)
telefono (cornetta, tastiera_o_rotella_numeri_da_0_a_9)
televisore (presa_antenna, pulsanti_dei canali, schermo, volume_luminosità_contrasto)
unità_centrale (cavo_alimentazione, floppy_o_hard_disk, memoria, microprocessore, schede_interfacce_o_espansione)

Ritorniamo al menu del programma gestore della base dati, CIBER00. L'opzione A (Aggiornamento della base-dati della conoscenza) serve ad aggiornare l'archivio contenente gli oggetti della base dati. Scelta l'opzione A apparirà un altro menu dal quale eseguiremo la scelta per visualizzare tutto l'archivio o un solo oggetto. Se vogliamo, per esempio, aggiungere o modificare degli attributi all'oggetto calcolatrice, dovremo digitare "calcolatrice", altrimenti, se vogliamo apportare delle modifiche sugli attributi di un oggetto del quale non conosciamo il nome dobbiamo digitare "*". Successivamente appariranno sullo schermo tutti gli oggetti presenti in archivio con a lato un numero identificatore. Per selezionare un oggetto sarà sufficiente inserire il numero che lo identifica. Quindi, dopo aver individuato un oggetto o per il suo nome oppure per il numero con cui è etichettato, passiamo ad un'altra schermata nella quale viene visualizzato un menu in basso al video. Tale menu è composto da tre opzioni:
D= cancello tutto l'oggetto
A= aggiorno oggetto
I= gestione attributi

Scegliendo l'opzione D si andrà a cancellare dalla base dati l'oggetto selezionato ed ovviamente anche tutti i suoi attributi. Con l'opzione A, invece, si potrà modificare il nome di tale oggetto senza però interferire negli attributi che lo definiscono. Infine con l'opzione I si potranno cancellare, modificare o inserire nuovi attributi circa l'oggetto selezionato. Scegliendo quest'ultima opzione apparirà una nuova schermata con un nuovo menu di possibili operazioni. Innanzi tutto con i tasti delle frecce su e giù si potranno visionare tutti gli attributi. Volendo inserire un nuovo attributo basterà premere il tasto INS e scrivere la nuova caratteristica dell'oggetto. Attenzione: gli attributi possono essere composti da più parole purché unite tra di loro dal segno "_". Volendo inserire l'attributo "è alto", per esempio, dovremo scrivere "è_alto". Nell'archivio si possono anche eliminare degli attributi, per far questo basta posizionarsi su quello che si vuole cancellare e premere il tasto DEL. Apparirà una "D" a sinistra dell'attributo quindi basta premere il tasto END e l'attributo verrà cancellato.

La SixPakPremium/EGA è una scheda da aggiungere al PC per aumentare le sue prestazioni. Espande la memoria a 2 Mb, supporta la grafica EGA, CGA, Hercules e monocromatica, incorpora una porta seriale ed una parallela nonché l'orologio calendario. Per informazioni: DHT, Milano, tel. 02/804168.

Come abbiamo detto l'archivio gestito dal programma CIBER00 rappresenta la base dati, o la conoscenza, utilizzata dal sistema esperto (programma CIBER51). Nel disco troverete già archiviati alcuni oggetti. Per aggiungere degli altri bisogna selezionare l'opzione A del menu principale, dopodiché scrivere il nome del nuovo oggetto (es.: rasoio_elettrico) e poi usare i comandi di gestione dell'archivio per inserire (tasto INS) gli attributi che definiscono il nuovo oggetto.

Dopo aver aggiornato, cancellato o inserito in archivio uno o più attributi, dobbiamo ritornare al menu principale e richiamare l'opzione D (preparazione del database della conoscenza) prima di lanciare il sistema esperto tramite l'opzione E. Questa procedura è estremamente importante: se non viene chiamata l'opzione D prima della chiamata del sistema esperto quest'ultimo non si troverà la base dati aggiornata e quindi non potrà considerare le ultime variazioni apportate sull'archivio.

Per sapere quali oggetti sono presenti in archivio basta usare l'opzione B nel menu principale mentre per stamparli bisogna usare l'opzione C.

L'accesso al sistema esperto avviene digitando l'opzione E del menu principale. Sul video apparirà una nuova schermata nella quale si vedranno comparire diverse domande da parte del computer. L'utente dovrà rispondere con "s" per sì o con "n" per no. Se vogliamo uscire dal programma dobbiamo inserire come risposta la parola "vedo". Il computer a questo punto formulerà lo stesso una possibile risposta circa l'oggetto da noi pensato e ci chiederà se vogliamo abbandonare il programma oppure no.

L'uso di questo sistema esperto può sembrare inizialmente utile solo a scopo didattico. In realtà, più si arricchisce la base dati della conoscenza e più ci si rende conto che il programma può essere utilizzato nella ricerca mirata di alcune strutture definite da dagli attributi, cioè da delle caratteristiche. Pensate per esempio di costruire una base dati dove l'oggetto è rappresentato da un tipo di malattia e gli attributi dai suoi sintomi. Se si riesce a definire questo archivio (cioè la conoscenza del sistema esperto) si ottiene un programma in grado di fare una diagnosi del malato in base ai suoi sintomi. Oppure provate a creare una

base dati sui minerali. Questi ultimi rappresentano gli oggetti mentre le caratteristiche di ognuno di essi i loro attributi. Provate poi ad usare il sistema esperto ed avrete un programma capace di dedurre il tipo di minerale in base alle conoscenze che ha nella base dati. Potremmo fare tanti altri esempi di utilizzo di un programma del genere, basta creare una buona base dati, dettagliata e precisa in continuo aggiornamento ed otterrete un programma davvero esperto, capace di aiutarvi nelle vostre ricerche. Non trascurate questo programma perché sarà tanto più utile quanto più voi sarete in grado di arricchire la base dati!

LE PAROLE RICONOSCIUTE

NOMI

casa
stanza
pavimento
porta
finestra
ingresso
bagno
pranzo
cucina
soffitto
terrazzo
sedia
letto
poltrona
tavolo
lampadario
padre
madre
mamma
babbo
radio
televisione

AVVERBI

sicuramente
luminosamente

VERBI

ha
hanno
giocano
abbiamo
dipinge
lavora
ascolta
ascoltano
guarda
guardano
chiede
pulisce
è
sono
gioco

ARTICOLI

il
lo
gli
la
le
un
una
uno

AGGETTIVI

grande
pulito
lucido
trasparente
caldo
freddo
acceso
spento
bello
bella
buono
buona
bravo
brava

PROPOSIZIONI

a
in
e
con

Dizionario delle parole riconosciute dal programma di elaborazione del linguaggio naturale.

NOTE PER I PROGRAMMATORI

Il programma CIBER00 è stato scritto in dBase III Plus e compilato in Clipper. I files sorgenti del programma sono registrati su disco con l'estensione .PRG. Consigliamo una loro visione utile per capire alcune tecniche di interfaccia tra utenti e programmi in Prolog per la gestione della base dati della conoscenza. Inoltre il programma è un'ottima dimostrazione didattica sull'interazione a livello funzionale tra linguaggi diversi e la possibilità di utilizzare il Prolog per funzioni "intelligenti" in procedure informatiche tradizionali.

Il programma gestisce due database; il primo, dal nome CIBER01, contiene il nome della base dati della conoscenza, knowledge, e il nome dell'oggetto da riferirsi. Per semplicità il programma è limitato al database "CIBER" ma ognuno può facilmente superare questa limitazione. Il secondo database è CIBER02 che ha i riferimenti tra l'oggetto e l'attributo. Il primo menu di CIBER00 fornisce la traccia per l'esecuzione dei vari comandi compreso il lancio del programma in Turbo Prolog compilato dal nome CIBER51. Infine viene usato un terzo database, CIBER03, che serve di appoggio per costruire il file CIBER 51.DAT di riferimento dati del programma in Turbo Prolog CIBER51.EXE.

RICONOSCITORE DI FRASI

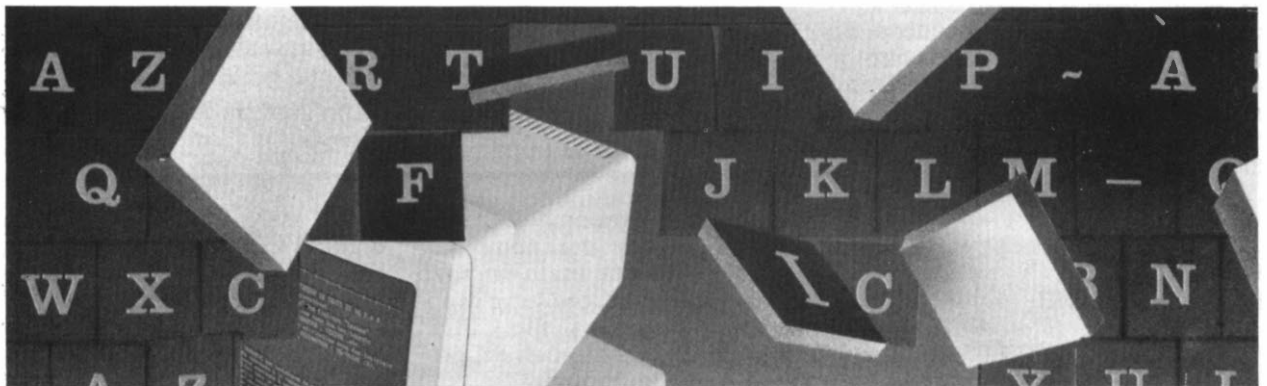
Un elaboratore di linguaggio naturale (ELN) è un programma che riceve in input una frase e l'analizza prima sintatticamente, poi cerca di dedurne il significato. Raggiungere questi risultati con un calcolatore vuol dire aprire nuove opportunità di lavoro con i computer. Pensate soltanto ad un nuovo sistema di consultazione dei dati basato sull'interrogazione in linguaggio naturale e non codificato, consentirebbe a chiunque l'uso del computer.

Questo programma rappresenta un esempio di ELN. La sua funzione si ferma all'analisi sintattica della frase ma essendo ben commentato può essere facilmente modificato per continuare anche nella possibile interpretazione della frase digitata.

Per usare il programma bisogna digitare CIBER60. La frase da analizzare deve contenere le parole riportate nell'elenco del dizionario e deve terminare con un punto ("."), inoltre tutte le parole devono essere scritte in minuscolo; quindi la frase non deve iniziare in maiuscolo. La grammatica usata, molto semplice, non prevede né la coniugazione dei verbi né l'uso di parole composte e neppure trasformazioni da singolare a plurale. Il programma contiene le (prime) regole di composizione grammaticale della frase ma, lo ripetiamo, non esegue alcun altro esame formale né sulle parole, né sul significato delle stesse.

Nel file CIBER60.PRO si trova il programma sorgente dell'elaboratore del linguaggio naturale scritto in Turbo Prolog. Esso è ampiamente commentato e questo lo rende facilmente leggibile ed interpretabile anche agli inesperti di programmazione in Prolog. Per coloro, invece, che vorranno ampliare le possibilità di questo programma non sarà difficile creare delle procedure di caricamento di un dizionario esterno indipendente e con possibilità di aggiornamento. ■

LA RICERCA DELLE SOLUZIONI

LE TECNICHE EURISTICHE APPLICATE ALLA INTELLIGENZA ARTIFICIALE PER RISOLVERE UN PROBLEMA. LA MASSIMIZZAZIONE DEI RISULTATI E LA MINIMIZZAZIONE DEGLI SFORZI.

La più importante applicazione dell'AI è la ricerca delle soluzioni. Esistono classicamente due tipi di problemi, il primo può essere risolto con una procedura deterministica che garantisce sempre il successo chiamata computazione, lavora bene solo in quei processi nei quali esistono dei calcoli come nella matematica.

Si può facilmente constatare che il metodo usato per risolvere questo tipo di problemi può essere facilmente tradotto in un algoritmo che un calcolatore può eseguire.

Nei programmi di tipo tradizionale è molto facile inserire delle istruzioni per risolvere tale tipo di problemi, putroppo sono poche le situazioni del mondo reale che portano a soluzioni di tipo computistico, le altre necessitano della ricerca della soluzione, ed è questo metodo che richiede le tecniche dell'AI.

Uno degli ostacoli più difficili da superare quando si tentano di applicare le metodologie di AI ai problemi del mondo reale è la gestione della dimensione e della complessità della maggior parte delle situazioni.

Quasi sempre la ricerca della soluzione di un problema inizia con un'analisi delle informazioni coinvolte e dei legami logici tra i dati del problema.

Qualche volta questi legami sono rappresentabili immediatamente con le strutture e gli algoritmi del linguaggio di programmazione usato; altre volte tali legami debbono essere rappresentati, con strutture più complicate.

Particolarmente frequente è il caso costituito dalle strutture ad albero.

Si dice albero una struttura costituita da uno o più nodi a livelli diversi, il nodo più alto si dice padre, quello più basso si dice figlio. Un albero ha queste caratteristiche:

- Ogni nodo ha, al massimo, un padre.
- Esiste un solo nodo, detto radice, che non ha padre.
- Nodi che non hanno figli si dicono foglie.

Una struttura ad albero può rappresentare una situazione reale e la ricerca di una soluzione del problema rappresentato può voler dire scorrere tutti i percorsi da un nodo a quello successivo.

I percorsi possibili più comunemente usati sono:

-Depth-first: consiste nell'andare dal nodo al primo figlio (a sinistra). Quando non ci sono più figli, o quando si arriva ad una foglia, il percorso risale fino a trovare il primo nodo già visitato che ha ancora figli da visitare.

Il processo termina quando il percorso torna alla radice e non ci sono più figli da visitare.

-Breadth-first: Viene realizzata col

visitare tutti i nodi di uno stesso livello (da sinistra a destra) prima di passare ai livelli del nodo successivo.
-Hill-climbing: La ricerca Hill-climbing sceglie come passo successivo il nodo che sembra più vicino al goal. Deriva il proprio nome da uno scalatore che si è perso nel buio, a mezza via verso una montagna; si assume che il campo dello scalatore sia sulla cima della montagna, allora, anche nella notte, lo scalatore pensa che ogni passo che va verso l'alto è nella giusta direzione.

Quest'ultima tecnica di ricerca della soluzione è veramente valida in molti casi dato che tende a ridurre il numero dei nodi che debbono essere visitati prima di arrivare alla soluzione, ma può soffrire per alcuni inconvenienti.

Il primo è quello delle false colline, il secondo è quello degli "altopiani" nei quali tutti i passi sono alla stessa altezza, il terzo è quello dei crepacci, ove lo scalatore cade ed è necessario che il processo di backtracking superi questa situazione durante il lavoro.

Nonostante questi aspetti, il processo di hill-climbing conduce normalmente a risultati più rapidi di ogni altro metodo non euristico.
-Least-cost: Minimizza lo sforzo per raggiungere l'obiettivo, sia economico che di percorso.

Schema di percorso Breadth-first.
Vengono attraversati tutti i nodi di un livello.

AGGIUNTA DELLE TECNICHE EURISTICHE

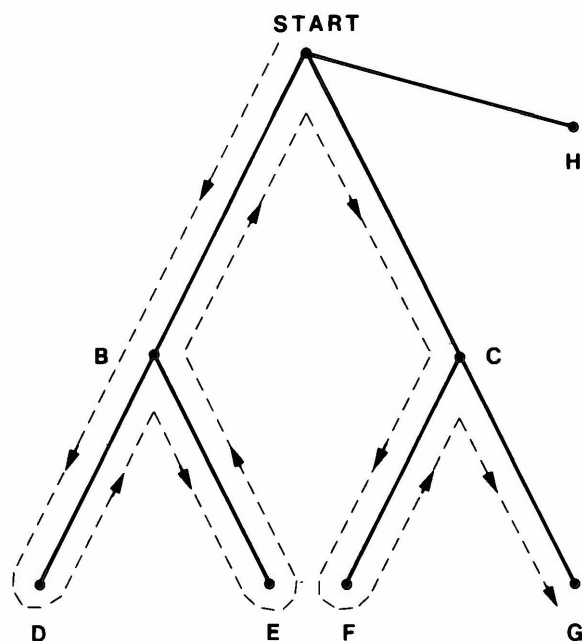
Le prime due tecniche di soluzione si possono assimilare al comportamento di un cieco che cerca la strada a tentoni, seguendo il percorso che si trova davanti. Ma questa ricerca di soluzione è tipica della programmazione classica, e l'AI deve trovare le proprie soluzioni in tempi mediamente migliori di quelli della normale programmazione.

È allora necessario aggiungere delle tecniche euristiche, così come ho definito il termine.

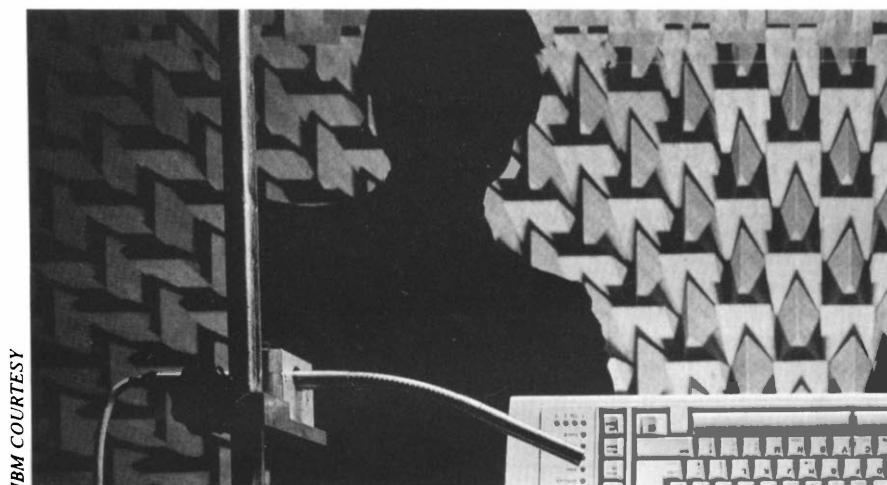
Supponiamo di dover trovare, durante una gita in un bosco ove abbiamo smarrito la strada, il ritrovo presso un torrente. Sappiamo che i torrenti sono normalmente nel fondo della valle, che gli animali tracciano dei sentieri per arrivare all'acqua, che possiamo sentire scorrere l'acqua quando siamo vicini e, infine, che possiamo sentire l'umidità quando siamo nei pressi del torrente.

Queste cognizioni pur non certe, ma euristiche, permettono di trovare rapidamente l'acqua, ma converrete con me che è impossibile generalizzarle a tutte le possibili contingenze.

Come elemento di lavoro vorrei dire subito che se si pensasse che sia facile inserire delle informazioni euristiche in un programma Prolog commetteremmo un errore, ma se non è possibile generalizzare è sicuramente possibile, come nei due casi sopra riportati, analizzare le situazioni che di volta in volta si presentano all'attenzione.

È chiaro che le tecniche euristiche, come nei due casi sopra riportati, tendono o a massimizzare i risultati o minimizzare gli sforzi.

Schema di percorso Depth-first.
Consiste nell'andare dal nodo al primo figlio.

I SISTEMI ESPERTI

COME IL CALCOLATORE USA LE INFORMAZIONI PER RICAVARE LE DEDUZIONI LOGICHE.

IBM COURTESY

Ciascuno di noi ritiene di essere, nel proprio settore, un esperto, in quest'ottica supponiamo di trovarci di fronte al problema di trasferire la nostra esperienza ad altre persone che dovrebbero o integrare il nostro lavoro o sostituirci in tutto o in parte. Anche una tipica attività professionale o consulenziale richiede un comportamento quasi mai ripetitivo, la nostra conoscenza viene sollecitata in modo casuale e colloquiale.

Se volessimo catturare la nostra esperienza e conoscenza per farci aiutare nel lavoro inserendola in un calcolatore per soddisfare le esigenze esposte nel paragrafo precedente, allora staremmo cercando di realizzare un sistema esperto.

Ma come realizzarlo, da dove possiamo cominciare?

Sul mercato esistono degli strumenti sviluppati da molte aziende che sono per l'esperto o per l'ingegnere della conoscenza come il foglio elettronico è per l'uomo d'affari o il tecnico.

Questi strumenti sono sia ambienti per lo sviluppo dei sistemi esperti che ambienti per la consultazione di sistemi esperti — sono delle conchiglie (shell), sistemi generalizzati per consultare e realizzare le applicazioni di sistemi esperti.

Usando strumenti di questo tipo è possibile per esperti inserire facilmente informazioni e creare un sistema esperto nel dominio di una particolare professione.

Gli ambienti di sviluppo e di consultazione dei sistemi esperti sono stati pensati dopo avere osservato che gli esperti o gli ingegneri della conoscenza spesso hanno difficoltà nella creazione di un sistema esperto e sembra possibile progettare un ambiente per aiutarlo.

Gli ambienti di sviluppo sono spesso progettati per fornire una base per la costruzione di una vasta gamma di specifiche applicazioni di sistemi esperti, gli ambienti di consultazione permettono all'utente di eseguire facilmente le applicazioni.

Si può pensare come a sistemi vuoti o conchiglie dove gli esperti o gli ingegneri della conoscenza inseriscono le proprie regole o conoscenze e scelgono il motore di deduzione (di inferenza), che sarà applicato alle regole.

Da questo deriva che un sistema esperto è, nella sua essenza, composto da due parti: la base della conoscenza e il motore di inferenza o di deduzione.

La base dati della conoscenza raccoglie le nozioni specifiche di un esperto e le regole di comportamento relative.

Questi due termini sono necessari per proseguire nel lavoro:

Oggetto: La conclusione che segue una regola.

Attributo: Una qualità che, assieme con la regola, aiuta a definire l'oggetto.

Possiamo pensare alla base della conoscenza come ad una lista di oggetti con i relativi attributi e regole.

Per molti sistemi esperti si possono definire gli oggetti come una lista di attributi che un oggetto possiede o non possiede. Esempio

Farfalla

ha le ali polverulente.
ha le antenne.
ha vita breve.
ha simpatia per i fiori.
ha corpo molle.
ha non le penne.

LA DEDUZIONE

Il motore di inferenza è quella parte del sistema esperto che tenta di usare le informazioni fornite per trovare gli oggetti che sono conformi alla base dati della conoscenza.

Esistono due categorie di sistemi esperti: quelli deterministici e quelli probabilistici. Per capire la differenza pensiamo a due esperti, un chimico ed un sociologo. Il chimico ci dirà con certezza che se un certo elemento ha due elettroni è quello che noi chiamiamo elio, mentre se chiediamo ad un sociologo di prevenire il fenomeno della droga, ci potrà suggerire dei comportamenti che hanno una certa probabilità di successo.

Tenendo conto delle due categorie dei processi decisionali, si possono identificare tre modi di lavorare di un motore inferenziale: catena in avanti, catena all'indietro e valorizzazione delle regole.

Il metodo con ragionamento in avanti è anche chiamato trainato dai dati poiché il motore di inferenza utilizza le informazioni fornite per muoversi attraverso una rete di AND e OR logici fino a raggiungere un punto terminale che è l'oggetto. Se il motore di inferenza non riesce a trovare un oggetto usando le informazioni esistenti, ne chiede altre all'operatore.

Le regole che definiscono l'oggetto creano i passi che conducono a lui, pertanto la sola strada per raggiungere l'oggetto è di soddisfare tutte le sue regole: un motore di inferenza con concatenamento in avanti parte con alcune informazioni e tenta di trovare un oggetto che soddisfa le informazioni ricevute.

Il metodo di concatenamento all'indietro è il contrario di quanto sopra detto, il motore di inferenza parte da una ipotesi (un oggetto) e chiede informazioni per confermare o negare l'ipotesi. Molti dei motori di inferenza forniti con il Turbo Prolog o con gli esempi da me pubblicati sono costruiti con la tecnica del concatenamento all'indietro.

Il metodo di valutazione è concettualmente superiore ai due precedenti, dato che richiede le informazioni che hanno maggior valore. La teoria generale delle operazioni è che il sistema analizza le informazioni che rimuovono la maggior incertezza dal sistema. ■

Esempio di ambiente di sistema esperto.
Sopra l'ingegnere della conoscenza e l'esperto del dominio sviluppano su calcolatore il sistema esperto; sotto l'utente consulta il computer come se fosse un esperto di settore.

CARO COMPUTER TI SCRIVO

NEL GERGO INFORMATICO SI CHIAMA "ELN" E STA AD INDICARE L'ELABORAZIONE DEL LINGUAGGIO NATURALE: IL LAVORO PIÙ IMPORTANTE NEL CAMPO DELL'AI. IL COMPUTER RIESCE A COMPRENDERE LE INFORMAZIONI SCRITTE IN NORMALE LINGUAGGIO UMANO!

Molti professionisti della A.I. pensano che il più importante lavoro che possono svolgere è quello relativo alla elaborazione del linguaggio naturale. La ragione per questo è che, una volta realizzata, la elaborazione del linguaggio naturale apre la porta direttamente al colloquio uomo macchina, ciò significa che una volta che una macchina potesse comprendere il linguaggio naturale, non sarebbero più necessari molti lavori programmati in modo standard.

Parola	*Tipo*
Porta	Nome
Casa	Nome
ha	verbo
corre	verbo
gioca	verbo
grande	aggettivo
elegantemente	avverbio
La	articolo
il	articolo
a	proposizione

Come vedremo in questo articolo, la elaborazione del linguaggio naturale è un lavoro da esperti, ma l'ampiezza e la complessità del linguaggio umano è ben lontana dall'essere completamente compresa. Spero di illustrare quanto possa essere semplice in certi casi la elaborazione del linguaggio, ma spero anche di trasmettervi quanto difficile sia far capire ad un calcolatore il significato di una frase.

Come detto in precedenza, la elaborazione del linguaggio naturale

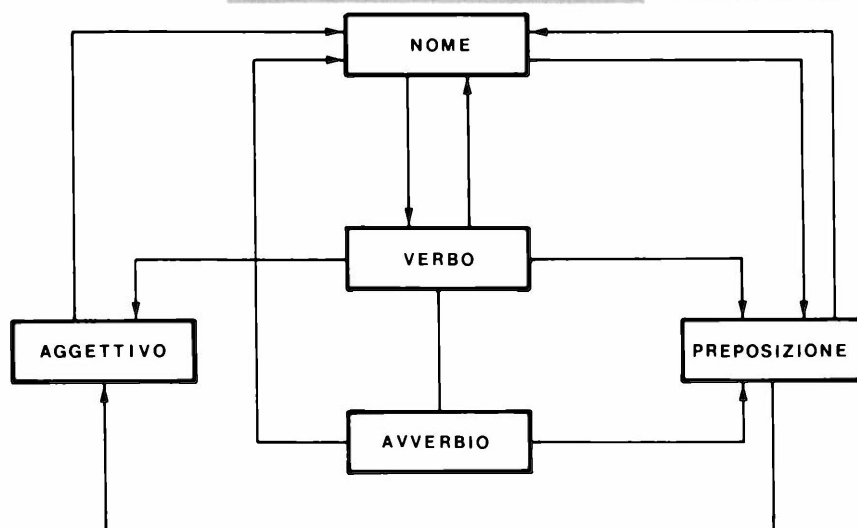



Figura 1. Schema dell'analizzatore di stato. In alto: Tabella 1. Esempio di dizionario riconosciuto nel sistema ELN.

```
/* definizione dei database */
   parola (symbol, symbol)   /*tipo, parola */
   stato  (symbol)         /* situazione corrente della  frase */
```

Figura 2. Definizione dei due database: quello delle parole e quello dello stato.

```
goal

/*carica un piccolo vocabolario */

 assert (parola(nome, porta)),
 assert (parola(nome, finestra)),
 assert (parola(proposizione, a)),
/* inizializzazione del database della situazione di stato */
 assert(stato(null)),
parti.
```

Figura 3. Sezione goal del programma. Carica il database "parola" ed inizializza il database "stato".

che d'ora in poi abbrevierò con ELN, tenta di rendere il calcolatore capace di comprendere dei comandi scritti in normale linguaggio umano. Userò chiaramente riferimenti alla dolce lingua del nostro paese, ma quanto andrò ad accennare si applica a tutte le lingue del mondo.

Un problema meno difficile e importante è quello di far rispondere in un linguaggio simile a quello umano; dopo che abbiamo reso il calcolatore capace di comprendere una domanda, è molto facile farlo rispondere.

Non sono d'accordo sul fatto che la sintesi della parola e il riconoscimento della voce siano parte della ELN, un programma non deve preoccuparsi come gli arrivano le frasi, questo è solo un problema tecnologico.

Una verità incontrovertibile è quella che una elaborazione del linguaggio naturale non è usabile da sola (al di fuori della ricerca), mentre deve fornire dei front-end per altri programmi, specialmente gestori di basi-dati risolutori generalizzati di problemi.

Possibili applicazioni delle tecniche ELN sono quelle rivolte all'interfacce per programmatori, per traduttori automatici e per l'automazione di robots.

IL NUCLEO DI UN ELN

Dato che il campo delle applicazioni del linguaggio naturale è veramente vasto, è difficile pensare di coprirlo in una pubblicazione come questa, io mi limiterò a far conoscere tre differenti aspetti dell'approccio alla elaborazione del linguaggio naturale.

Il nucleo di ogni sistema ELN è l'analizzatore, che è una sezione del

```
parti:-
 write("Digita la frase: "),
 readln(S),
 analizza(S),
 write ("Tutto bene la frase: "),nl,
 purge.
parti:- purge.
```

Figura 4. Struttura del predicato "parti". La sua funzione è quella di leggere una frase da tastiera e di analizzarla tramite la routine "analizza".

```
parola_dopo(S,S2,W):-
  trova_delim(S,Conta,0),!,
  Conta > 0,
  frontstr(Conta,S,W,S3),
  leva_spazio(S3,S2).
```

Figura 5. Predicato "parola_dopo". Serve ad estrarre da una frase una parola alla volta.

```
/* trova uno spazio o un punto *
trova_delim(S,conta,C):-
   frontchar(S,CH,S2),
   CH <> ' ' , CH <> '.',
   C2 = C + 1,
   trova_delim (S2,Conta,C2).
trova_delim(_,Conta,Conta).

leva_spazio(S,S2):-
  frontchar(S,Ch,S2),
  Ch = ' ' .
leva_spazio(,S).
```

Figura 6. Predicato "leva_spazio" usato per eliminare gli spazi all'interno della frase da analizzare.

```
/* controlla che ogni parola sia nel giusto ordine */
analizza(S):-
   parola_dopo(S, S2, W),
   stato(X),!,
   elabora(W,X),!,    /*  guarda  se ogni parola e'
   analizza(S2).      /*  al posto giusto */
analizza(S):-   /* controlla che ci sia il giusto terminatore */
   frontchar(S,CH,_),
   CH='.'.
```

Figura 7. Predicato "analizza". La sua funzione è quella di valutare la correttezza delle parole all'interno della frase da esaminare chiamando il predicato "elabora".

```
/* Elabora ogni nuova parola basata sullo stato corrente.
   elabora viene chiamata con la parola corrente e lo stato,
   cioe':   elabora(parola, stato).
   Se  la  parola e' corretta allora il nuovo stato e' messo  nel
   database di stato e la elaborazione continua     */

elabora (W,_):-
   parola (T,W),
   T = determiner,    /* non fare niente */
   !.
elabora(W,null):-
   parola(T,W),
   asserta(stato,(T)),!.
elabora(W,nome):-
   parola(verbo,W),
   asserta(stato,(verbo)),!.
elabora(W,nome):-
   parola(proposizione,W),
   asserta(stato,(proposizione)),!.
elabora(W,verbo):-
   parola(proposizione,W),
   asserta(stato,(proposizione)),!.
elabora(W,verbo):-
   parola(nome,W),
   asserta(stato,(nome)),!.
elabora(W,verbo):-
   parola(avverbio,W),
   asserta(stato,(avverbio)),!.
elabora(W,verbo):-
   parola(aggettivo,W),
   asserta(stato,(aggettivo)),!.
elabora(W,avverbio):-
   parola(proposizione,W),
   asserta(stato,(proposizione)),!.



elabora(_,_):-
  write ("Errore nella frase"),
  nl,!,fail.
```

Figura 8. Struttura del predicato "elabora". Viene chiamato da "analizza" e valuta, oltre alla parola, anche la sua corretta collocazione nella frase.

programma che legge realmente le frasi.

Ciascun analizzatore vede una frase in modo differente a seconda della propria applicazione.

Esistono due modi diversi per avvicinarsi alla elaborazione del linguaggio naturale; il primo che cerca di utilizzare i contenuti significativi di una frase (le informazioni), così come facciamo noi normalmente; il secondo cerca di far accettare alla macchina dei comandi in linguaggio naturale, ma solo estraendo la parte essenziale del comando, ciò è un compito più facile da gestire.

Io cercherò di illustrare un programma che cerchi di sviluppare quelle tecniche che possano convertire un linguaggio naturale in una forma comprensibile da un calcolatore.

Una delle maggiori difficoltà che si incontrano nel costruire dei sistemi di ELN è di restringere la complessità e la flessibilità del linguaggio umano nel sistema. Quando si implementa un elaboratore di linguaggio naturale, esiste una forte tentazione di restringere il tipo di frasi che il calcolatore possa comprendere ad un sottoinsieme del linguaggio naturale.

Se si restringe la grammatica ad un livello che il calcolatore possa accettare, il compito diventa più facile, e, se svolto correttamente, la restrizione è difficilmente rilevabile. In ogni caso io dovrò ricorrere a certe restrizioni per gli scopi di queste note, e assumerò, per la maggior parte degli esempi, che le frasi siano dichiarative e non interrogative.

Così che normalmente seguiranno la forma standard:
Soggetto, verbo, oggetto.

Assumerò anche che tutti gli aggettivi qualificativi seguiranno il sostantivo che modificano così come gli avverbi per i verbi. Poi assumerò che ogni frase termini con un punto.

Le frasi valide saranno di questo tipo:
La mamma va al mercato.
La vecchia signora veste elegantemente.

Mentre la frase
La vecchia signora è elegantemente vestita.
viene considerata non valida dal mio analizzatore; questa grammatica la chiamo G1, come molti altri autori in materia.

In aggiunta alle regole grammaticali di cui sopra, dovremo dotarci di un vocabolario, e per rendere i miei esempi semplici, vi consiglio di mantenere al minimo il numero delle parole usate, che l'analizzatore proposto riconoscerà come nell'esempio di tabella 1.

L'ANALIZZATORE DELLO STATO

Questo tipo di ELN usa la normale situazione delle frase per predire che tipo di parola dovrebbe trovarsi successivamente. La figura 1 indica l'analizzatore di stato che io userò.

Prima di poter implementare un programma che lavori secondo la gestione degli stati di lavoro, occorre definire due database. Il primo conterrà il vocabolario delle parole conosciute dal sistema e il loro tipo; il secondo conterrà lo stato corrente della frase in esame (vedi figura 2).

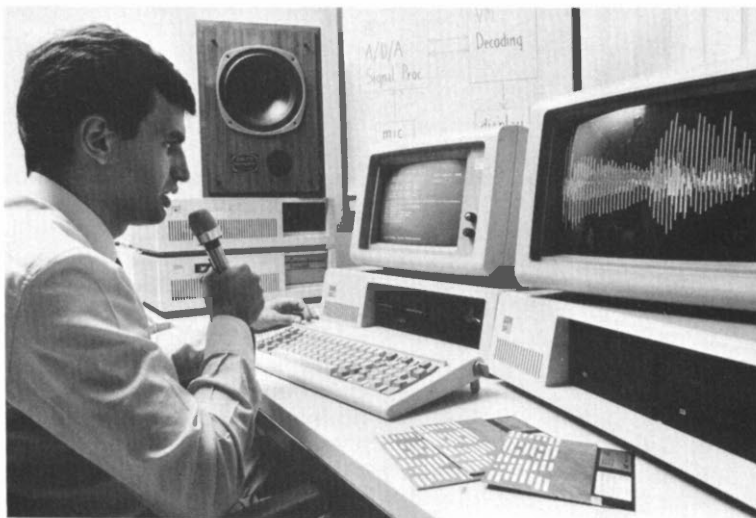
La sezione goal (vedi figura 3) del programma caricherà il database "parole" e poi inizializzerà il database di "stato" che conterrà il valore iniziale nullo.

Il predicato "parti" (vedi figura 4) legge una frase caricata dalla tastiera e chiama la routine "analizza" che spezza la frase nelle sue componenti.

L'IBM NELL'AI

Il Centro di Ricerca IBM di Roma ha messo a punto un sistema di riconoscimento automatico della voce. Il prototipo realizzato funziona nel seguente modo: tramite un microfono si trasmettono al computer delle onde vocali (parole e frasi) le quali vengono esaminate e convertite in segnali digitali, quindi tradotte in una sequenza numerica. Dopo questa fase di traduzione subentra un'elaborazione del riconoscimento acustico e linguistico basato sulle tecniche dell'intelligenza artificiale.

Sempre la IBM Italia ha messo a punto, in collaborazione con l'Ospedale San Raffaele, il sistema esperto Trace (Tutor for computer Aided Request for Clinical Exams) per assistere il medico nella scelta degli esami clinici.

Un altro sistema esperto messo a punto dalla IBM e realizzato dal laboratorio d'informatica Carisma di Perugia è CR.E.S. (Credit granting Expert System), utile agli operatori di banca con poca esperienza specialistica. Il sistema, infatti, sulla base dei dati forniti dal cliente della banca, suggerisce se sia il caso di emettere un finanziamento e a quali condizioni.

Per gl'istituti di assicurazione è nato ASS.I.RI, un prototipo sperimentato dalle Assicurazioni Generali. Questo sistema esperto consiglia la giusta copertura assicurativa a seconda dei dati forniti dal cliente.

L'informatica, con l'intelligenza artificiale, può entrare anche in tribunale. Il Sistema Esperto Legale (SEL) sviluppato dalla IBM per conto della Corte d'Appello di Roma mette a disposizione dei magistrati un supporto informativo e decisionale nel campo dei Processi Civili.

Se la routine "analizza" ritorna "true(vero)" la frase è corretta ed in accordo con la grammatica ridotta, se ritorna "false" allora la frase non è corretta.

Il predicato "purge" pulisce il database dopo il termine della elaborazione.

Prima di sviluppare l'analizzatore per la gestione dello stato, bisogna creare pochi predicati che divideranno la frase nelle parole individuali. Ogni frase è rappresentata come una stringa di caratteri. Così viene dichiarato un nuovo tipo di dato, la "frase" nella sezione dei "domains".

domains
 frase = string

Sfortunatamente il predicato contenuto nel Turbo Prolog "fronttoken" non permette abbastanza flessibilità in certe situazioni, come una parola divisa in sillabe, dunque dovremo scrivere delle routine per farlo. Dato che vogliamo estrarre tutte le parole della frase una alla volta, creiamo un predicato da chiamare per primo la cui struttura è visibile nella figura 5.

Si tratta di "parola_dopo", viene chiamato con tre argomenti: S contiene la frase corrente, S2 sarà riempito di ciò che rimane in S dopo che la prima parola della frase è rimossa e W conterrà la parola rimossa.

Nel predicato "parola_dopo", come vedete, ho usato il predicato del Turbo Prolog "frontstr" che fornisce solo i primi N caratteri; dunque sarà necessario che "parola_dopo" calcoli la lunghezza della prima parola usando "trova_delim", che conta i caratteri fino a quando non trova uno spazio o un punto.

L'array Motorola MCA 1500 M: logica ad alta velocità (300 picosecondi) per 1500 gates equivalenti, più 1552 bit di RAM configurabili da 3-5 ns, organizzata in quattro blocchi di 32x9. Insomma hardware sempre più potente per maggiori sviluppi dell'AI.

Vorrei ricordare che nella nostra grammatica le parole sono delimitate solo dallo spazio o dal punto.

Il predicato "leva_spazio" (figura 6) toglie gli altri spazi della frase.

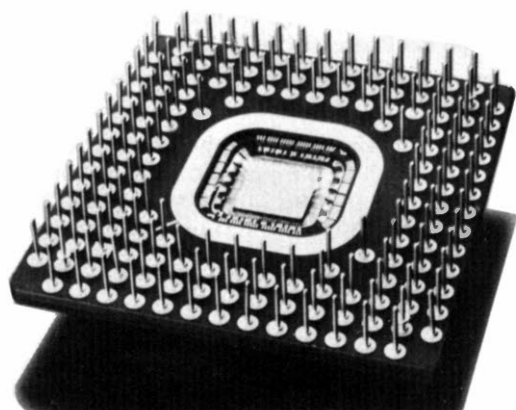
Dopo aver visto i predicati di supporto, passiamo a descrivere il predicato di analisi (figura 7).

Come potete vedere "analizza" è ricorsivo; ogni volta che è chiamato, viene levata una parola e lo stato è determinato con la linea "stato(X)", che unitamente alla parola corrente, è passato al predicato "elabora" che effettua il cambiamento di stato e controlla la correttezza di ciascuna parola in relazione al corrente stato (vedi figura 8).

Il predicato "elabora" lavora in questo modo: se la parola successiva nella frase è diversa da un verbo, un avverbio un nome, un aggettivo o una proposizione, la prima clausola ha successo e non avviene la transazione di stato. La seconda clausola fa partire la conversione di stato. Dato che il database di stato contiene inizialmente solo uno stato nullo e a causa della seconda clausola, la prima parola dalla frase farà sì che elabora tornerà vero. Le rimanenti clausole, escluso l'ultimo, sono usate per eseguire le corrette transazioni di stato. Se nessuna altra clausola ha successo, allora viene eseguita l'ultima clausola che scrive un errore e poi termina.

Ho volutamente non terminato la descrizione delle elaborazioni, completate nel programma allegato; invito il lettore a confrontare le proprie soluzioni.

ELN PER TESTO LIBERO

Per capire un analizzatore di testo libero bisogna pensare alla costruzione della frase in modo completamente diverso rispetto al modello precedente, cioè bisogna pensare ad una frase composta di elementi, che a loro volta sono composti di altri elementi, e così via fino ad un livello atomistico, cioè ai verbi, nomi, aggettivi, ecc. Le regole che governano la costruzione di ogni parte del programma sono chiamate "regole di produzione". Un analizzatore a testo libero di frasi usa queste regole di produzione per analizzare una frase.

IN LIBRERIA

INTRODUZIONE AI SISTEMI ESPERTI

Peter Jackson
Masson
pp. 312 — 39.000 lire
testo in lingua italiana

I sistemi esperti sono programmi che lavorano su una base dati, o conoscenza, e dalla quale riescono a trarre deduzioni in aiuto al lavoro umano in un certo campo, dalle diagnosi cliniche alle prospezioni geologiche. Di fatto, i sistemi esperti, rappresentano l'applicazione dell'intelligenza artificiale attualmente più significativa.
Il volume parte dall'analisi dei sistemi esperti e della loro collocazione nell'ampio settore dell'intelligenza artificiale. Viene poi esaminata la ricerca euristica, componente fondamentale di un sistema esperto.

STORIA DELL'INTELLIGENZA ARTIFICIALE

Pamela McCorduck
Franco Muzzio Editore
pp. 418 — 42.000 lire
testo in lingua italiana

Questo libro è indicato a tutti coloro che desiderano una lettura divertente. L'autrice, infatti, racconta i tentativi più strani, misteriosi, comici e non di costruire dei sistemi di intelligenza artificiale basati su delle macchine. Il libro tratta in particolar modo il mondo che ha vissuto queste insolite esperienze.

HEURISTICS
Intelligent search strategies for computer problem solving.
Judea Pearl
Addison Wesley
pp. 382 — 105.000 lire
testo in lingua inglese

Le applicazioni informatiche dell'intelligenza artificiale portano all'adozione della programmazione di tipo euristico, cioè l'algoritmo non è più basato su formule computistiche ma su strategie deduttive. Questo libro si suddivide in tre parti. La prima affronta alcuni esempi di giochi per introdurre il concetto di risoluzione tramite tecniche euristiche e loro rappresentazione grafica. Continua con la descrizione delle procedure basilari di ricerca euristica (hill-climbing, depth-first, backtracking, ecc.) per poi passare alla descrizione di una loro ottimizzazione e ad un confronto strategico del loro utilizzo. La seconda parte del libro affronta dal punto di vista matematico alcuni problemi applicando le tecniche e le procedure euristiche descritte nella prima parte del libro. Nella trattazione ricorrono spesso confronti e giudizi descritti matematicamente con rappresentazione grafica. Infine l'ultima parte del libro tratta la decisione strategica e deduttiva nei giochi con l'implementazione della ricerca probabilistica.

INTRODUZIONE AL TURBO PROLOG

Carl Townsend
Franco Muzzio Editore
pp. 320 — 35.000 lire
testo in lingua italiana

Questo libro è un valido strumento di lavoro con il quale sperimentare le tecniche di intelligenza artificiale in linguaggio Turbo Prolog della Borland. Ricco di esempi e programmi, la sua stesura è stata studiata in particolar modo per coloro che intendono autoapprendere questo linguaggio.

A GUIDE TO EXPERT SYSTEMS

Donald A. Waterman
Addison Wesley
pp. 420 — 76.000 lire
testo in lingua inglese

Il libro tratta in sei sezioni il campo dei sistemi esperti: una delle aree dell'intelligenza artificiale maggiormente applicate anche in campo industriale. Il volume incomincia con la definizione di sistema esperto, di come è strutturato, quali sono le differenze tra un programma convenzionale ed un sistema esperto ed analizza alcune possibili strutture di sistemi esperti applicati ai differenti campi della scienza (dalla elettronica alla medicina). La seconda sezione del libro raccoglie in capitoli separati una trattazione più approfondita delle componenti di un sistema esperto e delle loro relazioni soffermandosi sulla citazione di sistemi esperti già realizzati. Le due sezioni successive del volume esaminano le problematiche inerenti alla costruzione e allo sviluppo di un sistema esperto considerando l'acquisizione delle informazioni e la loro messa a punto.

PROGRAMMARE IL PROLOG PER L'INTELLIGENZA ARTIFICIALE

Ivan Bratko
Masson
pp. 416 — 42.000 lire
testo in lingua italiana

Il Prolog è un linguaggio dedicato alla trattazione dei dati non numerici e particolarmente indicato per i lavori di intelligenza artificiale. Il libro rappresenta un'opera completa su questo linguaggio adottato anche nei calcolatori giapponesi della quinta generazione.

IL COMPUTER CHE PENSA

Bertram Raphael
Franco Muzzio Editore
pp. 446 - 38.000 lire
testo in italiano

Può il computer pensare? Questo volume è una completa introduzione all'intelligenza artificiale, scritta da

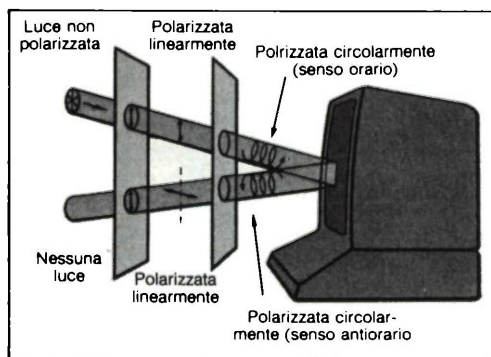
uno dei ricercatori padre di questa nuova scienza informatica. Come introduzione, il libro analizza in profondità i concetti fondamentali e le idee di base della IA anziché considerare i casi delle applicazioni. L'opera è completa di una presentazione accurata delle tecniche principali di soluzione dei problemi.

I testi qui presentati possono essere richiesti a: Hoepli (tel. 02/865446) o al Centro Editoria Informatica (tel. 02/77971).

Polaroid protegge i tuoi occhi

I filtri Polaroid sono gli unici con polarizzatore circolare

POLAROID è la più qualificata specialista nel trattamento della luce ed è quindi naturale che abbia risolto al meglio i problemi degli operatori di terminali video.

Problemi causati dal riverbero della luce ambiente e da mancanza di contrasto sullo schermo, che possono generare bruciore agli occhi, mal di testa, vertigine.
Esistono sul mercato alcuni filtri che eliminano il riverbero, altri che migliorano il contrasto.
I filtri POLAROID ottengono entrambi i risultati grazie, soprattutto, al loro esclusivo polarizzatore circolare che intrappola la luce ambiente riflessa dallo schermo e contemporaneamente eliminano lo sfarfallio dei caratteri e li rende più nitidi e meglio leggibili.
Prodotti in cristallo o poliestere, con o senza messa a terra, i filtri POLAROID sono disponibili in varie dimensioni per meglio adattarsi ad ogni terminale.
E per gli schermi curvi tipo Olivetti, esistono appositi adattatori stampati in ABS.

Quando la luce ambiente si riflette sullo schermo viene intrappolata dal polarizzatore circolare inserito nel filtro Polaroid e non ritorna più agli occhi dell'operatore. Mentre la luce emessa dallo schermo attraversa il filtro depurata da aloni e sfarfallii e con un contrasto enfatizzato.